DATEL Electronics

L i m i t e d

EUROSYSTEMS

AMIGA ACTION REPLAY II

INSTRUCTION MANUAL

by Olaf Boehm and Joerg Zanger

I N H A L T S V E R Z E I C H N I S

Vorwort

Wir danken Ihnen für Ihr Vertrauen, das Sie uns, dem Hause EUROSYSTEMS und unserem Produkt AMIGA ACTION REPLAY, entgegengebracht haben.

Wir arbeiten fortwährend an der Verbesserung und Ergänzung von AMIGA ACTION REPLAY. Ein preisgünstiger Update-Service ist bei Ihrem Kauf übrigens inbegriffen (sofern die Hardware nicht verändert wird). Wir werden Sie rechtzeitig über neue Versionen von AMIGA ACTION REPLAY in Fachzeitschriften informieren. Sollten Sie Verbesserungsvorschläge oder Ideen für spätere Versionen von AMIGA ACTION REPLAY haben, senden Sie diese doch an uns!

Unsere Adresse:
EUROSYSTEMS
HÜHNERSTR. 11
4240 EMMERICH
TEL.: 02822/45589 oder 45923

Übrigens, wenn Sie AMIGA ACTION REPLAY im vollen Umfang nutzen wollen, sollten Sie folgende Einleitung und die Anhänge aufmerksam durchlesen. Sie enthalten viele nützliche Tips für den Umgang mit AMIGA ACTION REPLAY.

Mit freundlichem Gruß

EUROSYSTEMS

Allgemeine Worte zu AMIGA ACTION REPLAY

Wer kann AMIGA ACTION REPLAY gebrauchen?

AMIGA ACTION REPLAY wurde so programmiert, daß gleichermaßen Anfänger und Profis unterstützt werden. Sind sie Anfänger, so werden Sie mit AMIGA ACTION REPLAY mit der Zeit zum AMIGA Profi. Der Profi dagegen kann nun mit AMIGA ACTION REPLAY die volle Systemkontrolle über seinen Amiga erhalten, wie es sonst nicht möglich ist. Für Programmentwickler ist es nun einfacher, Ihr Programm auf Fehler zu untersuchen.

Was kann AMIGA ACTION REPLAY ?

Sicherlich ist dieses eine der ersten Fragen, die Sie sich stellen. Wir möchten Ihnen nun einen kleinen Auszug aus den Funktionen von AMIGA ACTION REPLAY geben.

1. Freezen von Programmen ohne Rücksicht auf einen Kopierschutz, d.h. der Zustand des Amigas wird vollständig auf Diskette abgespeichert. Das so abgespeicherte Programm kann dann zu einem späteren Zeitpunkt eingeladen werden. Sie können nun normal weiterarbeiten. Auf diese Art sind Sie nun in der Lage, sich eine Sicherheitskopie von Ihrem Orginalprogramm zu machen, ohne sich mit verschiedenen Kopierprogrammen ärgern zu müssen.
2. Trainen von Programmen, d.h. Sie können nun in fast jedes Spiel/Programm Ihren persönlichen Trainer einbauen, ohne Spezialkenntnisse in Maschinensprache zu besitzen. Vorbei ist die Zeit, in der Sie in Fachzeitschriften nach einem Trainer für ein bestimmtes Programm suchten. Ab heute können Sie nun selber in Fachzeitschriften Ihre Trainer anbieten.
3. Bilder abspeichern, d.h. Sie können nun in fast allen Programmen das aktuelle dargestellte Bild abspeichern und dieses z.B. mit einem normalen Malprogramm weiterbearbeiten oder einfach eine "DIA-SHOW" aus solchen Bildern zusammenstellen. Sie können natürlich auch das abgespeicherte Bild mit einem üblichen Druckprogramm ausdrucken u.s.w.
4. Das Programm analysieren, d.h. mit vielfältigen Befehle ist es nun dem einfachen Anwender möglich, sich ein Programm anzuschauen und das Programm z.B. zu verändern oder aus dem Programmierstil zu lernen.
5. Abspeichern von Sounds, d.h. mit einem einfachen Befehl ist es Ihnen nun möglich, aus einem Programm heraus den gerade gespielten Sound abzuspeichern und ihn z.B. in ein Musikprogramm einzuladen oder anderweitig zu verwenden.
6. Erkennen/Vernichten von Viren, d.h. immer, wenn Sie in den Modul-Editor gehen, wird Ihnen angezeigt, ob sich möglicherweise ein Virus im Speicher des Amigas festgesetzt

hat. Mit AMIGA ACTION REPLAY hat man nun auch die Möglichkeit, beim Reset jeden Virus zu löschen und dabei falls ein Virus gefunden wurde, ein Warnzeichen abzugeben.

7. Abschaltung einer Speichererweiterung, d.h. falls ein Programm mit einer Speichererweiterung nicht laufen sollte, können Sie mit einem einfachen Befehl Ihre Speichererweiterung ausschalten und wieder einschalten.
8. Automatisches Erkennen und Unterstützung eines zweiten Laufwerkes.
9. Automatisches Erkennen und Unterstützung einer internen 512 KB Speichererweiterung.
10. Einfach zu bedienender Spriteeditor, d.h. mit einfachen Befehlen können Sie nun Sprites wie z.B. den Mauszeiger verändern.
11. Vielzahl von Befehlen, um den Speicher bzw. das Programm zu untersuchen und zu verändern.
12. Serienmäßig eingebaute Bremse, d.h. falls Ihnen ein Programm zu schnell sein sollte oder Sie das Programm analysieren wollen, können Sie nun einfach mit einem Drehregler die Geschwindigkeit des Amigas stufenlos auf eine Ihnen genehme Geschwindigkeit - bis fast zum Stillstand - herabsetzen.
13. Eingebaute Pausenfunktion, d.h. wenn Sie in einem Programm eine Verschnaufpause brauchen, das Programm aber keine Pausenfunktion hat, können Sie nun einfach durch Drücken des Modul-Editor-Tasters eine beliebig lange Pause einlegen.
14. Eingebauter Bildschirm-Editor mit zwei voneinander unabhängigen Bildschirmen.
15. Eingebauter deutscher Zeichensatz.

Installation von AMIGA ACTION REPLAY in Ihrem Amiga

Zum Lieferumfang von AMIGA ACTION REPLAY gehören:

1 Modul AMIGA ACTION REPLAY
1 deusches Handbuch
1 Referenzkarte

Achten Sie darauf, daß Ihr Amiga und sämtliche angeschlossenen Geräte bei der Installation ausgeschaltet sind. Falls Sie das Modul bei eingeschalteten Geräten einstecken, sind Schäden am Modul und an Ihren Geräten wahrscheinlich.

Einbau im Amiga 500/1000:

Entfernen Sie nun eine eventuell vorhandene Abdeckung vorsichtig

von Ihrem Expansions-Port. Dieser befindet sich beim Amiga 500 auf der linken, beim Amiga 1000 auf der rechten Gehäuseseite.
Halten Sie das AMIGA ACTION REPLAY jetzt so, daß die Aufschrift "AMIGA ACTION REPLAY" nach oben zeigt und schieben Sie das Modul in dieser Stellung vorsichtig auf Ihren Expansions-Port. Achten Sie darauf, daß das Modul gerade und fest im Expansions-Port steckt. Drücken Sie das Modul ruhig etwas fester hinein.
Nachdem Sie sich vergewissert haben, daß das AMIGA ACTION REPLAY richtig im Expansions-Port steckt, können Sie nun Ihre Amiga normal einschalten. AMIGA ACTION REPLAY ist nun betriebsfertig.

Update-Service

Selbstverständlich arbeitet unser Entwicklungs-Team ständig an Verbesserungen von AMIGA ACTION REPLAY. Durch unseren preisgünstigen Update-Service können Sie auch künftig immer auf dem neuesten Stand von AMIGA ACTION REPLAY sein. Wird jedoch die Hardware vollständig verändert, ist leider kein Update von Ihrer älteren Version mehr möglich.

Über die Erscheinung der neuen Versionen werden wir sie rechtzeitig in den Fachzeitschriften informieren. Um unseren Update-Service in Anspruch zu nehmen, schicken Sie uns einfach Ihr "altes" Modul. Gegen einen Unkostenbeitrag wird Ihr Modul aufgerüstet und wird Ihnen wieder zurückgeschickt. Dieses gilt jedoch nur - wie schon vorher erwähnt - wenn die Hardware nicht verändert wird. Unsere Adresse finden Sie im Vorwort!

Handhabung der Bremse

Die Bremse, die Ihren Amiga nahezu stufenlos fast bis zum Stillstand abbremsen kann, kann zu jeder Zeit ein- und ausgeschaltet werden. Den augenblicklichen Zustand der Bremse erkennen Sie an der roten Leuchtdiode. Leuchtet diese, so ist die Bremse aktiv. Mit dem Drehregler können Sie die Geschwindigkeit, mit der Ihr Amigaprogramm arbeitet, stufenlos regeln (von voller Geschwindigkeit bis zu 20 %). Die Bremse wird automatisch, während Sie im Modul-Editor arbeiten, ausgeschaltet, so daß sämtliche Cartridge-Befehle immer mit maximaler Geschwindigkeit abgearbeitet werden.

ACHTUNG: Bei manchen Programmen ist es möglich, daß diese mit eingeschalteter Bremse nicht mehr einwandfrei laufen. Man sollte dann die Bremse abschalten!

Befehlseingabe

Haben Sie den Modul-Editor-Taster gedrückt, erscheint der Editor-Bildschirm mit weißer Schrift auf blauem Grund (falls nicht von Ihnen geändert mit dem COLOR-Befehl oder der Preferences-Seite). Außerdem erscheint ein blinkender Cursor (weißer Unterstrich "_"). Sie können jetzt mit der Befehlseingabe beginnen.

Falls Sie einmal einen Befehl vergessen haben, betätigen Sie einfach die Help-Taste. Es wird dann eine Liste aller Befehle und Editor-Tasten ausgegeben - zur Gedächnisstütze. Zur Befehlseingabe stehen Ihnen folgende Editor-Tasten zur Verfügung:

F1	löscht den aktuellen Bildschirminhalt
F2	setzt den CURSOR in die linke obere Bildschirmecke
F3	Preferences-Menü, siehe Preferences
F5	gibt den gesamten Modul-Bildschirm über einen evtl. angeschlossenen EPSON-kompatiblen Drucker aus
F6	schaltet das Printerprotokoll aller Befehlsein- und ausgaben ein/aus
F7	wählt zwischen den Modi Überschreiben und Einfügen
F8	gibt die Tastaturbelegung/Sonderfunktionen während des P-Befehls aus (MemoryPeeker)
F9	schaltet zwischen deutschem und amerikanischem Tastaturtreiber um
F10	wechselt zum zweiten Hilfsbildschirm
<SHIFT> F10	wechselt zum ersten (normalen) Eingabebildschirm
HELP	gibt einen Hilfstext aus
SHIFT	stoppt das Bildschirmrollen, so daß Sie sich alles in Ruhe ansehen können
TAB	fügt ein Leerzeichen ein
DEL	entfernt das Zeichen unter dem CURSOR
BACKSPACE	entfernt das Zeichen vor dem CURSOR

Die vier CURSOR-Tasten werden auch unterstützt. Betätigt man diese mit der SHIFT-Taste, erfüllen Sie auch Sonderfunktionen, wie in erste/letzte Zeile springen und zum nächsten/vorherigen Wort springen.

WICHTIG**WICHTIG**WICHTIG**WICHTIG**WICHTIG**WICHTIG**WICHTIG
Viele Befehle wie z.B. der Speicherdump, der Disassembler, ASCII-Ausgabe u.a., bieten die folgende komfortable Möglichkeit:
Sind einmal Informationen (z.B. Speicherinhalte oder beim Disassembler Maschinenbefehle) oben aus dem Bildschirm herausgerollt, so können diese durch die Cursor-Hoch-Taste wieder nach unten gerollt werden, falls der Cursor sich in der obersten Bildschirm-Zeile befindet. Die jeweiligen Adressen werden dann entsprechend erniedrigt !!

Bitte drücken Sie nun einmal den Modul-Editor-Taster und probieren Sie die Editortasten aus.

Beachten Sie, daß jede Befehlszeile mit einem RETURN abgeschlossen sein muß.
Während der Befehlsabarbeitung blinkt die Power LED des Amigas. Fast alle Befehle lassen sich mit der ESC-Taste abbrechen.

Zahleneingabe

Beim ACTION REPLAY AMIGA können Sie sämtliche Zahleneingaben bei Befehlen hexadezimal, dezimal oder binär angeben. Dabei interpretiert das Modul "normal", d.h. ohne besondere Kennzeichnung eingegebene Zahlen, als Hexadezimalzahlen. Wenn Sie z.B. die Zahl "12" eingeben, interpretiert dies das Modul als Hexadezimalzahl mit dem Wert (dezimal) !18. Um ausdrücklich anzugeben, daß es sich bei einer eingegebenen Zahl um eine Hexadezimalzahl handelt, kann der Zahl auch ein Dollarzeichen "$" vorangestellt werden: $12.

Wollen Sie eine Zahl dezimal interpretiert wissen, muß diese durch ein vorangestelltes Ausrufungszeichen "!" gekennzeichnet werden, z.B. "!32" oder "!65535". Binäre Zahlen dagegen markieren Sie mit einem vorangestellten Prozentzeichen "%", z.B. "%1001" oder "%0011110101011".
So werden also folgende Zahlen vom Modul als gleich angesehen:

!128 = $80 = 80 = %10000000
%01001010 = 4A = !74 = $4A

Führende Nullen können bei der Adressen- und Zahleneingabe weggelassen werden!
z.B.: $00052340 = 52340

Hexadezimalsystem:

Das hexadezimale Zahlensystem stellt Zahlen zur Basis 16 dar, während im Dezimalsystem mit der Basis 10 gerechnet wird. Im Hexadezimalsystem sind daher Buchstaben für die Zahlen 11-15 nötig. Es werden dafür die Buchstaben "A" bis "F" verwendet.
Es gilt folgende Übersetzungstabelle:
A = !10, B = !11, C = !12, D = !13, E = !14, F = !15

Beispiel:
hex to dez: $FE = $15*16^1 + 14*16^0$ = !254
$3E8 = $3*16^2 + 14*16^1 + 8*16^0$ = !1000

Der Calculate Befehl nimmt automatisch alle notwendigen Umwandlungen vor und gibt das Ergebnis in allen Zahlensystemen aus!
z.B.: ? $1000 - $100 + !256 - %1 = $fff

PREFERENCES:

Wenn Sie im Modul-Editor anstatt einen Befehl einzugeben die Taste F3 drücken, kommen Sie auf die Preferences-Seite. Mit der linken Maustaste können Sie nun die einzelnen Felder (= Gadgets) anklicken. Ein aktiviertes Feld wird durch eine diverse Farbe dargestellt. Verlassen können Sie die Preferences-Seite durch das Anklicken von "USE" oder durch das Drücken der ESC-Taste. Es werden aber auf jeden Fall die geänderten Einstellungen übernommen.

Nun die Bedeutung der anklickbaren Felder im einzelnen.

Links oben: MEMORYCONTROL

Durch Anklicken von den "FAST-Gadgets" können Sie Ihr Fastmem - falls vorhanden - an- und abschalten. Mit dem "CHIP-Gadget können Sie - falls möglich - zwischen 1 MB und 512 kB Chipmem wählen. Durch das Anwählen des "CLEAR-RAM-Gadget" wird das Löschen des Speichers beim Reset eingeschaltet.
Die geänderte Speicherkonfiguration wird erst beim nächsten Reset aktiv.

Links Mitte: DRIVECONTROL

Mit den "ON/OFF-Gadgets" können Sie Ihre Diskettenlaufwerke abschalten (Laufwerk df0 darf bei Kick 1.2/1.3 nicht abgeschaltet werden).
Die "BOOTSEL-Gadgets" bestimmen, von welchem Laufwerk beim nächsten Reset gebootet werden soll.

OFF:	Normales Booten von Laufwerk df0
VAR:	Booten von jedem Laufwerk, indem eine Diskette beim Reset eingelegt ist. Die Priorität der Laufwerke beträg dabei df3 > df2 > df1 > df0.
df0-df3:	Booten von dem gewählten Laufwerk

Links Unten: COLOR

Durch diese Gadgets können Sie Ihre Farben im Modul selber einstellen.
Mit "BACKGND" und "FOREGND" und den Farbwerten können Sie die Farbe der Zeichen und des Hintergrundes frei wählen.

Mitte oben: VIRUSTEST

Durch Einschalten des Virustestes wird:

- bei jedem Reset der Speicher auf Viren untersucht
- bei jedem Aufruf des Moduls der letzte Virus beim Reset angezeigt und gleichzeitig wird der Speicher nach neuen Viren durchsucht
- vor dem Einladen des Bootblocks einer Bootdiskette eine Kopie der Exec-Base und anderen wichtigen Register erstellt. Nach dem Ausführen des Bootblocks wird dann auf verdächtige Veränderungen getestet und gegebenenfalls ins Modul gesprungen, damit der Anwender in einem erweiterten Virenmenü verschiedene Möglichkeiten hat.

Rechts oben: DAUERFEUER

Beim Anklicken der verschiedenen Prozentmarken (0%-100%) kann den Joysticks/Mäusen ein Dauerfeuer simuliert werden, wobei 100% für die größtmögliche Schußfrequenz und 0% für manuelles Bedienen des Feuerknopfes steht. Bei manchen Programmen kann aber die maximale Frequenz schon niedriger als bei 100% liegen.

Mitte unten: USE

Beim Anklicken von "USE" wird die Preferences-Seite verlassen und die Einstellungen übernommen.

<u>Beschreibung der Modul-Editor-Befehle</u>

Zum besseren Verständnis der folgenden Befehlsbeschreibung erläutern wir nun einige wichtige und oft gebrauchte Fachbegriffe:

Syntax

Die Syntax eines Befehles ist die genaue Beschreibung eines Befehles und seiner möglichen Parameter.

Parameter

Ein Parameter ist ein Zahlenwert oder ein String, welcher an einem Befehl angehängt wird.

Adresse

Eine Adresse bedeutet einen Zeiger (Zahl) aus den physikalischen Speicher (RAM/ROM) des Amigas. Um Fehlbedienungen zu erschweren, werden nur folgende Adressbereiche als gültig anerkannt:

$000000 - $100000	: Chip-RAM (bis max. 1 MB)
$200000 - $400000	: externe Speichererweiterungen
$500000 - $A00000	: -"- -"-
$C00000 - $C80000	: interne 512 KB Speichererweiterung
$C00000 - $D00000	: interne 1 MB Speichererweiterung
$C00000 - $DC0000	: interne 1.8 MB Speichererweiterung
$DC0000	: Echtzeituhr
$FC0000 - $FFFFFF	: Kickstart-ROM

Zahlenwert

Ein Wert ist eine Zahl zwischen $0 und $ffffffff, d.h. ein ganze Zahl ohne Vorzeichen.

String

Ein String ist eine Zeichenfolge, eingeschlossen von Anführungszeichen (können bei AmigaDOS-Dateinamen weggelassen werden). Wenn ein String verlangt wird, kann dieser aber auch mit Daten gemischt werden. z.B.:

1. "DIES IST EIN STRING"
2. "DIES" $20 "IST" $20 "AUCH EIN STRING"
3. $20 $21 $22 $45 $56

Alle drei Beispiele stellen Strings dar, allerdings mit unterschiedlicher Länge und Inhalt.

PC

Bei dieser Bezeichnung handelt es sich um die Abkürzung für (P)rogram-(C)ounter. Der PC enthält die Adresse des Maschinenbefehls, den der Prozessor (beim Amiga ein M68000) als nächstes ausführen würde.

Task

Ein Task ist nichts anderes als ein Program. Da der Amiga das "Multitasking" beherrscht, schaltet er schnell zwischen internen Tasks (= Programmen) um. Davon merkt der Anwender allerdings nichts und glaubt, daß alle Programme gleichzeitig ablaufen.

CPU

Der Begriff "CPU" ist die Abkürzung für (C)entral-(P)rocessing-(Unit). Dahinter verbirgt sich nichts anderes als der Mikroprozessor des Computers (beim Amiga 500 & 2000 ein M68000).

ACHTUNG:

Wenn bei den nachfolgenden Befehlsbeschreibungen in der Syntax ein Parameter in runden Klammern steht, so kann er weggelassen werden!

Wenn in der Syntax das Oder-Zeichen "¦" steht, gibt es mehrere Auswahlmöglichkeiten, aber nur eine von ihnen darf angegeben werden.

Wenn in der Syntax ein Parameter nicht in runden Klammern steht, muß er angegeben werden.

Freezer- und Ripper-Befehle

SA - Befehl Freezen auf Diskette

Syntax: SA (path)name(, crunchrate)

speichert das aktuelle Programm unter den Namen "name" im Verzeichnis "path" oder dem aktuellen Verzeichnis (s. CD - Befehl) ab. Es werden dabei automatisch alle wichtigen Daten auf Ihre AmigaDOS-Diskette abgespeichert. Vor dem Speichern der Daten werden diese vom Modul komprimiert (erkennbar am Bildschirmflackern). Die Effizienz des Packers kann mit der eventuell angegebenen "crunchrate" beeinflusst werden. Dabei wird um so besser gepackt (dauert aber länger!), je größer die angegebene Zahl (immer zwischen $10 und $7fff) ist. Standardmäßig eingestellt ist $20.

Bsp.: SA "0:game" - speichert auf Lauferk df(0) ab
SA 0:game - ebenso
SA spiel,30 - effektiverer Packvorgang

SR - Befehl Freezen auf Diskette/Modul verlassen

Syntax: SR (path)name(, crunchrate)

gleiche Funktion wie der SA - Befehl, jedoch wird nach Beendigung des Abspeichervorgangs automatisch das Modul verlassen und das abgebrochene Programm fortgesetzt. (Funktion wie SA - Befehl mit anschließendem X - Befehl)

Bsp.: siehe SA -Befehl

LA - Befehl Einladen eines gefreezten Files

Syntax: LA (path)name

lädt das gefreezte File "name" aus dem Verzeichnis "path" (oder dem aktuellen) von Diskette ein. Das so geladene Programm kann dann mit dem X-Befehl gestartet werden.

Bsp.: LA 2:ordner/game - lädt vom df(2) aus dem Verzeichnis "Ordner" das gefreezte File "game".

LR - Befehl Einladen und Starten eines gefreezten Progammes

Syntax: LR (path)name

wie LA-Befehl, jedoch wird nach dem Einladen und Entpacken

des Programms dieses sofort gestartet. (Funktion wie LA- Befehl mit anschließendem X-Befehl)

Bsp.: siehe LA - Befehl

SLOADER - Befehl Abspeichern des Ladeprogramms

Syntax: SLOADER

speichert ein vom AmigaDOS CLI aus ladbares Programm ab (in das aktuelle Verzeichnis siehe CD - Befehl), mit dem Sie vom Modul gefreezte und abgespeicherte Programme auch ohne Modul wieder einladen können. Insbesondere können Sie dieses Ladeprogramm, das unter den Namen "ALOAD" abgespeichert wird, z.B. in das C: Verzeichnis Ihrer Festplatte kopieren (erst auf Diskette und dann vom CLI aus mit dem COPY-Befehl auf Ihre Festplatte).

Um ein Programm dann ohne Modul zu laden, müssen Sie von einem AmigaDOS-CLI aus 'ALOAD name' eingeben, wobei Sie für "name" den Namen (mit Pfad) des abgespeicherten Programms eingeben.

Bsp.: SLOADER - Speichert den Autoloader in das aktuelle Verzeichnis

Um ein abgefreeztes Programm zu laden und zu starten (auch ohne Modul) vom CLI aus, muß eingeben werden: ALOAD name

SQ -Befehl Abspeichern auf die RAM-Disk

Syntax: SQ

speichert den aktuellen Programmstand nicht auf Diskette, sondern in den (unbenutzten) Speicher Ihres Amigas, wobei ein eventuell zuvor (mit SQ-Befehl) abgespeichertes Programm überschrieben wird. Dies geht wesentlich schneller als das Abspeichern auf Diskette (SA-Befehl), hat jedoch den Nachteil, daß der gespeicherte Spielstand nach Abschalten des Computers verloren ist.

ACHTUNG: Dieser Befehl funktioniert nur, wenn Sie einen Amiga besitzen, der intern (!) über mindestens 1 MB RAM verfügt (z.B. A2000 oder A500 + interne Speichererweiterung) und dem Amiga beim Booten jedoch nur 512 KB zur Verfügung stellt. Dies erreichen Sie mit Hilfe der Speicherabschaltung der Preferences-Seite des Moduls (F3-Taste)!

Bsp.: SQ

SQR - Befehl Abspeichern in RAM und starten

Syntax: SQR

wie SQ-Befehl, startet jedoch nach dem Abspeichern das Programm wieder automatisch (Funktion wie SQ-Befehl mit anschließendem X - Befehl)

Bsp.: SQR

LQ - Befehl Einladen vom RAM

Syntax: LQ

lädt ein Programm vom unbenutzten RAM Ihres Amigas wieder ein, welches dann mit dem X-Befehl gestartet werden kann. Zuvor muß jedoch ein solches Programm mit Hilfe des SQ- oder SQR-Befehls abgespeichert werden.

Bsp.: LQ

LQR - Befehl Einladen und Starten vom RAM

Syntax: LQR

wie LQ - Befehl, jedoch wird nach dem Einladen des abgespeicherten Programms dieses gestartet. (Funktion wie LQ - Befehl mit anschließendem X - Befehl)

Bsp.: LQR

TRACKER - Befehl Sucht nach Musikstücken (Soundtracker)

Syntax: TRACKER (start)

durchsucht das ChipRAM (evtl. ab der mit "start" angegebenen Adresse) des Amigas nach Musikstücken im Soundtracker-Format. Wird der Befehl fündig (während der Suche blinken die Zeichen auf dem Schirm), so steht Ihnen ein weiteres Menü zur Verfügung:

F1 - spielt das gefundene Stück

F2 - stoppt das Abspielen wieder

F3 - gibt Informationen über die benutzten Instrumente und Samples, wie z.B. Name, Länge usw.

F4 - dient zum Abspeichern des Musikstückes auf Diskette. Nach Drücken dieser Taste müssen Sie den Namen des Stückes eingeben (evtl. mit Pfad), unter dem es auf Diskette gespeichert werden soll (+ RETURN). So abgespeicherte Musikstücke können dann mit den meisten üblichen Soundtrackern wieder eingeladen und weiterverwendet werden

F6 - gibt während dem Abspielen des Stücks die aktuell gespielten Musikdaten aus

F7 - setzt die Suche nach weiteren Musikstücken fort

F8 - verändert das Musikstück so, daß es auch mit Soundtrackern wieder eingeladen werden kann, die nur 16 Instrumente verarbeiten können

F9 - korrigiert eventuell falsch im Speicher stehende Pattern-Längen, so daß das Musikstück vom Modul aus richtig abgespielt und abgespeichert wird

F10 - beendet den Befehl und sie befinden sich wieder im Modul-Editor

Am besten funktioniert der TRACKER-Befehl mit Musikstücken aus INTROS und DEMOS, da diese oft standardisierte Musikdaten benutzen, die vom Modul gelesen werden können, aber auch bei einigen Spielen kann er die Musik finden.

Bsp.: TRACKER

SCAN - Befehl Suchen nach digitalisierten Samples

Syntax: SCAN

stellt das ChipRAM des Amigas als Klangkurve dar und stellt mittels eines Menüs weitere Möglichkeiten zur Verfügung:

Cursor - (rechts/links) verschieben der Start- und Endadresse des abzuspielenden Samples

Leertaste - schaltet zwischen Start und Ende um, den Werten, die dann mit Cursor rechts/links verändert werden können.

F1 - spielt den Sample vom eingestellten Start bis zum Ende mit der eingestellten Frequenz (Sampleperiod) ab.

F2 - zoomt den eingestellten Bereich graphisch, so daß dieser detaillierter betrachtet werden kann

F3 - setzt die Parameter Start, Ende und Sampleperiod wieder auf die anfänglichen Werte zurück

F4 - vergrößert den dargestellten (graphischen) Bereich, um mehr Übersicht zu erlangen (aus-zoomen)

F5 - speichert den eingestellten Sample als IFF-8SVX File ab, das von Musikprogrammen wieder eingeladen werden kann, die dieses Standard-Format verstehen.

1-4 - mit den Zahlentasten 1, 2, 3 und 4 werden die Parameter Start, Ende und Sampleperiod auf die Werte des aktuell gespielten Samples des Kanals 1-4 gestellt. Damit können Sie diese Sounds besonders schnell finden (oft durch aus-zoomen nach drücken der Zahlentasten)

Bsp.: SCAN

SP - Befehl Abspeichern von Bildern im IFF-Format

Syntax: SP (path)name(,picnr (höhe))

speichert das aktuelle Bild unter dem Namen "name" auf das verzeichnis "path" (oder das aktuelle) ab. Bei manchen Bildern, oft auch bei Spielen, kommt es vor, daß das Bild sich aus mehreren Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Diese können nur einzeln gespeichert werden. Welches Bild man nun speichern möchte,

gibt man mit "picnr" an. Läßt man die "picnr" weg, wird das erste Bild abgespeichert.

Falls die Bildhöhe vom Modul nicht richtig erkannt worden sein sollte oder mit Hilfe des P-Befehls eine eigene gewählt wurde, kann diese mit Bildhöhe angegeben werden.
Das abzuspeichernde Bild sollte vorher mit dem P-Befehl begutachtet werden!
Das Bild, welches der P-Befehl darstellt, wird 1:1 auf die Diskette abgespeichert werden!
Bilder, die im "Dual-Playfield-Modus" dargestellt werden, einem Amiga-eigenen Darstellungsmodus, bei dem zwei unterschiedliche Bildschirme transparent übereinandergelegt werden können, werden die in zwei einzelnen übereinandergelegten Bilder unterteilt und getrennt abgespeichert, was im Datei-Namen durch ein angehängtes ".1" oder ".2" (für den ersten und zweiten Bildschirm) kenntlich gemacht wird.

Bsp.: SP "1:bild",1 !200 - Speichert unter dem Name "bild" im Laufwerk df(1) das erste Bild mit der Bildhöhe von 200 Pixeln ab.

P - Befehl Suchen/Darstellen von Bildern

Syntax: P (picnr)

stellt das aktuelle Bild dar. Bei manchen Bildern, oft auch bei Spielen, kommt es vor, daß das Bild sich aus mehreren Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Diese können nur einzeln angezeigt werden. Welches Bild man nun sehen möchte, gibt man mit "picnr" an. Während der Bilddarstellung kann nun mit der rechten und linken Maustaste der untere Rand eingestellt werden. Bei Verlassen der Bilddarstellung mit der ESC-Taste wird dann die Höhe des so gewählten Bildes ausgegeben. Dieser Wert sollte beim Abspeichern des Bildes auch angegeben werden. Durch die Angabe der Bildhöhe beim Speichern haben Sie Gewissheit, daß nur das Bild abgespeichert wird und keine überschüssigen Daten. Außerdem stehen Ihnen während der Bilddarstellung noch folgende Befehle auf Tastendruck zur Verfügung:

a - autoplane:	setzt Planepointer mit Offset
b - brightness plus:	hellt das Bild auf
B - brightness minus:	dunkelt das Bild ab
c - colorreg plus:	Farbregister plus 1
d - dual playfield on:	Dualplayfieldmodus einschalten
D - dual playfield off:	Dualplayfieldmodus ausschalten
e - right border plus:	rechter Rand plus 1
E - right border minus:	rechter Rand minus 1
f - fast plane up:	ein Bild nach oben blättern
F - fast plane down:	ein Bild nach unten blättern
h - hold and modify on:	HAM - Modus ein
H - hold and modify off:	HAM -Modus aus
i - invert all colors:	invertiert alle Farbregisterwerte
l - lores mode on:	niedrige Aufslösung (40 Zeichen) ein
L - lores mode off:	hohe Auflösung (80 Zeichen) ein

m - modulo plus:	erhöht Modulowerte um 2
n - modulo minus:	erniedrigt Modulowerte um 2
s - left border minus:	linker Rand minus 1
S - left border plus:	linker Rand plus 1
w - white helpscreen:	weiße Parameterleiste
W - black helpscreen:	schwarze Parameterleiste
x - colorreg minus:	Farbregister minus 1
y - switch diw arid ddf mode:	Wähle Scrollmodus
+ - add plane:	erhöht die Planezahl
- - sub plane:	erniedrigt die Planezahl
0 - unlock all planes:	gibt alle Planes frei
SHIFT 0 - lock all planes:	wählt alle Planes fest (nur auf dem 10er-Block!)
1-6 unlock plane:	wählt jeweilige Plane fest
SHIFT 1-6 unlock plane:	gibt jeweilige Plane frei (nur auf dem 10er-Block!)
7 - red color plus:	
8 - blue color plus:	
9 - green color plus:	erhöht Farbwert des eingestellten Farbregisters
SHIFT 7,8,9:	erniedrigt Farbwert des eingestellten Farbregisters (nur auf dem 10er-Block!)
CURSORTASTEN:	rotieren/scrollen das Bild in die jeweilige Richtung
SHIFT UP/DOWN:	scrollen das Bild schneller hoch und runter
DEL - hide helpscreen:	schaltet die Parameterleiste ab
HELP - show helpscreen:	schaltet die Parameterleiste ein
F1 - default colors:	setzt Standardfarben
F10 - random colors:	setzt Zufallsfarben

Mit der Maus kann die eventuell eingeschaltete Parameterleiste beliebig auf dem Bildschirm postiert werden.
Um das jeweils auf dem Schirm angezeigte Bild abzuspeichern verlassen Sie den P-Befehl mit der ESC-Taste und benutzen Sie den SPM-Befehl (siehe unten).

SPM - Befehl Verändertes Bild abspeichern

Syntax: SPM (path)name

---

falls Sie mit Hilfe des P-Befehls das aktuelle Bild verändert haben, können Sie das VERÄNDERTE Bild mit Hilfe des SPM-Befehls unter dem Namen "name" abspeichern.

Bsp.: SPM ordner/test - Speichert das Bild aus dem Mempeeker (P -Befehl) unter dem Namen "test" im Directory "ordner" auf das aktuelle Laufwerk ab.

<u>Trainer - Befehle</u>

TS - Befehl Starte Trainvorgang

Syntax: TS wert

startet den Train-Modus und beginnt die Suche nach dem angegebenen Wert. Der angegebenen Wert stellt den Zählerstand dar, nach dem Sie suchen.
Mit dem TS/T - Befehl können Sie die Adresse des von Ihnen gesuchten Zählers erhalten. Mit dieser Information ist es Ihnen dann möglich, z.B. mit dem M-Befehl sich nach Bedarf mit Leben zu versorgen.
Wenn Sie einmal den Trainer gestartet haben sind Sie im "Trainmode".

Bsp.: TS !3
FIRST TRAINPASS!
CHANGE COUNTVALUE NEXT TIME!
SEARCHED UP TO: 005444
TRAINMODE AKTIVE!
READY.
X

T !2
SEARCHED UP TO: 080000
00014428
TRAINMODE AKTIVE
READY.

In diesem Beispiel hatten Sie am Anfang 3 (Leben), danach verließen Sie das Modul (X-Befehl) und verloren (absichtlich) 1 (Leben) und hatten nun nur noch 2 (Leben). Der Modul-Trainermaker hat nun als Addresse des Zählers 0014428 ausgegeben. Verändern Sie nun die bei Ihnen natürlich andere Adresse mit dem M-Befehl (im Bsp.: M 14428 + RETURN) und tragen Sie den neuen Wert (z.B. 09) ein. Nun verlassen Sie das Modul abermals und erfreuen sich Ihrer 9 (Leben) Oder aber fahren Sie fort wie folgt:

TX
TRAINMODE INAKTIVE!
TFD 14428
SUB FOUND AT 001122
SUBS ELIMINATED!

Um sich beliebig viele (Leben) zu verschaffen. Zur genauen Handhabung des Trainers siehe auch Anhang A "Handhabung des Trainers".

T - Befehl Trainvorgang fortsetzen

Syntax: T wert

setzt den Trainvorgang fort mit dem angegebenen Wert. Adressen, die verdächtig sind der gesuchte Zähler zu sein, werden ausgegeben. Wenn der Trainer nichts gefunden hatte, wird die Meldung "TRAINERMAKER WAS NOT SUCCESSFULL!" ausgegeben. In diesem Fall starten Sie bitte einen neuen Trainvorgang oder beenden Sie den Trainer mit dem TX-Befehl.

Bsp.: siehe TS-Befehl

TX - Befehl Trainvorgang beenden

Syntax: TX

Beendet den Trainermodus

Bsp.: siehe TS-Befehl

TF - Befehl Absoluter Trainer (beliebig viele Leben)

Syntax: a) TF adresse
b) TFD adresse

zu a)
zeigt die Adressen des Programmes an, wo wahrscheinlich mit dem Befehl "SUBQ.X #X,adresse" oder "SUBI.X #X,adresse" der Inhalt der angegebenen Adresse verringert wird (findet auch indirekte Adressierungsarten).

zu b)
wie a) aber zusätzlich werden diese Maschinenbefehle aus dem Programm entfernt -> Das laufende Programm zieht von dem in der Adresse enthaltenen Zähler nichts mehr ab!

Bsp.: siehe TS-Befehl

PC - Befehl Messe Energiebalken aus

Syntax: siehe P-Befehl

stellt das aktuelle Bild dar. Bei manchen Bildern, oft auch bei Spielen, kommt es vor, daß das Bild sich aus mehreren Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Diese können nur einzeln angezeigt werden. Welches Bild man nun sehen möchte, gibt man mit "picnr" an. Läßt man "picnr" weg, wird das erste Bild dargestellt. Während der Bilddarstellung können Sie mit Hilfe der Maus einen Bildausschnitt wählen, dessen Höhe und Breite in Bildschirmpunkten nach dem Drücken der ESC-Taste ausgegeben wird. Damit ist es leicht, die Länge von Energiebalken auszumessen und diesen Wert im Zusammnenhang mit dem Trainer weiterzuverwenden (Zähler).
Drückt man die linke Maustaste, kann man den linken oberen Rand des Auschnittes verschieben. Drückt man die rechte Maustaste, kann der rechte untere Rand verschoben werden.
Um für den Trainer verwertbare Resultate zu erzielen, sollte man darauf achten, den Bildauschnitt "pixelgenau" zu setzen!

Bsp.:
```
P 1
PICTUREHEIGHT: !256
READY

PC 1
WIDTH  : X = !123
HEIGTH: Y = !010
READY
```

Anti - VIRUS - Befehle

Nachfolgend folgen zwei Befehle zur Bekämpfung von Viren im Speicher des Computers. Dies ist aber nur der eine Teil eines wirksamen Virusschutzes. Der wichtigere Teil ist, Viren auf infizierten Disketten aufzuspüren und diese (die Viren) zu löschen. Das Aufspüren von Viren nimmt Ihnen von nun an das Amiga Action Replay MK II ab: Sobald Sie Ihren Computer einschalten, überwacht das Modul (sofern diese Option nicht mit Preferences ausgeschaltet wurde) sämtliche Disketten, von denen Sie booten und meldet Aktivitäten, die auf Viren schließen lassen.
Die nachfolgenden Befehle ergänzen lediglich dieses Programm und werden zum Teil auch automatisch ausgeführt.

VIRUS - Befehl Viren im Computer aufspüren

Syntax: VIRUS

durchsucht den Speicher nach Viren und zeigt diese an. Dieser Befehl wird vom Modul automatisch beim Aufruf mittels des Modul-Tasters ausgeführt, falls die VIRUSTEST-Option im Preferences-Menü angewählt wurde (beim Einschalten automatisch aktiviert).

Bsp.: VIRUS

KILLVIRUS Viren im Computer vernichten

Syntax: KILLVIRUS

sucht und entfernt Viren aus dem Speicher des Computers. Diese Maßnahme entfernt keine Viren auf Disketten. Um Viren von infizierten Disketten zu entfernen siehe INSTALL-Befehl.

Bsp.: KILLVIRUS

Disketten und Diskettenkodier - Befehle

Mit dem Amiga Action Replay MK II können Sie jetzt direkt mit AmigaDOS-kompatiblen Disketten arbeiten. Sämtliche Diskettenoperationen liefern direkt Dateien im AmigaDos-Format, manche auch im standartisierten IFF Format (Bilder, Sounds)

Neben dieser Möglichkeit können Sie jetzt aber auch Ihre Disketten mit Codewörtern vor fremdem Zugriff schützen (auch Sicherheitskopien u.ä.). Dazu bietet Ihnen das Amiga Action Replay MK II zwei verschiedene Möglichkeiten:

1.) Bei Programmen, die direkt vom Bootblock aus gestartet werden (z.B. Elite) und bei denen der übrige Disketteninhalt

in einem Fremdformat gehalten wurde (Not A DOS-Disk) , besteht die Möglichkeit den Bootblock durch ein Passwort zu schützen. Benutzen Sie dazu die Befehle BOOTCODE, BOOTPROT.

2.) Bei Programmen, die normal von AmigaDOS geladen werden können (z.B. SSI-Programme), kann die gesamte Diskette kodiert werden. Dazu muß die unkodierte Diskette in einem speziellen Verfahren umkopiert werden. Benutzen Sie dazu die Befehle CODE, CODECOPY.

WICHTIG: Arbeiten Sie beim Kodieren IMMER mit Sicherheitskopien !!!!!

BOOTCODE - Befehl Bootblockkodierer aktivieren

Syntax: BOOTCODE (codenumber)

stellt die Kodezahl (0 - $ffffffff) codenumber für den Bootblockkoder ein oder gibt die aktuelle Kodezahl aus, falls keine angegeben wurde. Ist die Kodezahl ungleich null, ist der Bootblockkodierer aktiviert und Sie können jetzt auch von kodierten Disketten booten (Unkodierte Disketten können Sie auch jetzt noch normal booten).
Um Ihre Disketten zu schützen, müssen Sie diese aber auch zuvor mit dem BOOTPROT-Befehl kodiert haben (s. unten)!

Bsp.: BOOTCODE 873233

aktiviert den Bootcoder und setzt das Kodewort auf die Zahl $873233 -> nur noch unkodierte und mit dieser Zahl kodierte Disketten können gebootet werden.

BOOTCODE 0

schaltet den Bootcoder wieder aus -> nur noch unkodierte Disketten können gebootet werden.

BOOTPROT-Befehl Kodiert den BootBlock einer Diskette

Syntax: BOOTPROT (codenumber)

kodiert den BootBlock der Diskette im aktuellen Laufwerk mit dem Kodewort "codenumber" (Zahl zwischen 0 und $ffffffff). Von dieser Art behandelten Disketten können Sie nur noch booten, wenn Sie mit Hilfe des Moduls das BootBlock - Kodewort auf diese spezielle Zahl einstellen (mit BOOTCODE-Befehl). Falls Sie die Kodezahl "codenumber" weglassen, wird die momentan eingestellte (BOOTCODE-Befehl) verwendet.
Um Disketten wieder zu decodieren, wenden Sie den BOOTPROT-Befehl mit Ihrem speziellen Kodewort, mit dem Sie sie kodiert haben, einfach nochmals auf die Diskette an.

Bsp.: CD 0: - wählt Laufwerk df(0) als aktuelles Laufwerk
BOOTPROT 12348765

Diskette im Laufwerk df0: ist kodiert und kann nicht mehr gebootet werden.

BOOTCODE 12348765

Modul wurde auf richtiges Kodewort eingestellt. Der Computer kann jetzt die immer noch kodierte Diskette wieder booten. Ohne Modul ist dies nicht mehr möglich.

BOOTPROT 13248765

Diskette ist wieder dekodiert und kann normal geladen werden.

CODE - Befehl Kodiert Diskettenlaufwerke

Syntax: CODE (drive (codenumber))

kodiert ein bestimmtes Laufwerk (drive) mit der Kodenummer "codenumber" (Zahl zwischen 0 und $ffff), so daß dieses Laufwerk ab sofort nur noch mit dieser Kodenummer kodierte Disketten akzeptiert. Falls keine Kodenummer angegeben wurde, wird das Laufwerk "drive" wieder in den normalen (unkodierten) Zustand versetzt. Die eingestellten Kodewörter bleiben auch über einen Computer-Reset hinaus erhalten, so daß Sie also auch von kodierten Disketten booten können.

ACHTUNG: Kodierte Laufwerke verarbeiten nur noch kodierte Disketten korrekt. Sämtliche Diskettenoperationen auf unkodierte Disketten von kodierten Laufwerken aus führt zu Fehlermeldungen.

Bsp.: CODE 0 1234 für das Kodieren und Verwenden von Disketten

Internes Laufwerk akzeptiert nur noch mit 1234 kodierte Disketten (auch das Modul).
Zu kodierende Disk in z.B. Laufwerk df1 einlegen.
Leere Disk in Laufwerk df0 (internes Laufwerk) einlegen
CODECOPY 1 0
Jetzt ist die Originaldiskette kodiert auf der zweiten abgespeichert worden.
Die kodierte Kopie kann nun nur noch auf Laufwerken betrieben werden, bei denen mittels des CODE-Befehls die Laufwerkkodiernug auf 1234 gesetzt worden ist.

CODE 0

schaltet die Kodierung des internen Laufwerks aus -> Die Kopie ist unlesbar geworden, das Laufwerk kann wieder normal genutzt werden u.s.w.

DCOPY - Befehl Kopieren von AmigaDOS-Disketten

Syntax: DCOPY source dest

Kopiert die Diskette vom Laufwerk "source" (0-3) auf die Zieldiskette im Laufwerk "dest" (0-3). Auch das Kopieren mit

nur einem Laufwerk ist möglich. Beachten Sie, daß der DCOPY- Befehl nur für unkodierte Disketten gedacht ist. Die Kodierung von dem Quell- und/oder Ziellaufwerk wird für die Dauer des DCOPY-Befehls ausgesetzt.

Der DCOPY Befehl arbeitet IMMER mit VERIFY, d.h. die Zieldiskette wird auf Schreibfehler untersucht und diese werden gegebenenfalls entfernt. Ebenso werden Lesefehler auf der Quelldiskette erkannt und wenn möglich korrigiert. Somit lassen sich auch widerspenstige und sogar defekte Diskette noch komfortabel verwenden oder sogar reparieren.

Bsp.: DCOPY 0 1

Kopiert Diskette vom internen Laufwerk auf das Laufwerk df1.

CODECOPY - Befehl Kodieren von Disketten

Syntax: CODECOPY source dest

kopiert die (gegebenfalls kodierte) Diskette (AmigaDOS) im Laufwerk "source" (0-3) auf die Diskette im Laufwerk "dest" (0-3) und wird dabei mit dem Kodewort des Laufwerks "dest" kodiert. D.h. wenn Sie z.B. eine Diskette auf das Kodewort 1234 kodieren wollen, kodieren Sie zunächst das Ziellaufwerk (dest) mit dem Befehl CODE dest 1234 (dest=0-3) und geben dann ein: CODECOPY source dest (source, dest=0-3). Die Diskette im "dest"-Laufwerk ist dann die kodierte Kopie des Originals im "source"-Laufwerk. Der Befehl funktioniert nur mit zwei Laufwerken!

ACHTUNG: Sollten Sie das Codewort vergessen oder verlieren, kommen Sie nicht mehr an die Daten heran. Verwahren Sie also das unkodierte Original sorgfältig.

CD - Befehl Wechsle aktuelles Verzeichnis

Syntax: CD (path)

setzt den aktuellen Modul-Pfad auf das Verzeichnis "path". Dabei gelten dieselben syntaktischen Regeln wie bei normalen AmigaDOS-Pfaden. EINZIGE AUSNAHME: Anstatt df0: geben Sie 0: ein. Anstatt df1: 1: u.s.w. Falls Sie keinen Pfad angeben wird der aktuelle Pfad ausgegeben.

Bsp.: CD 1:C

wechselt den aktuellen Pfad zum C: Verzeichnis der Diskette im Laufwerk df1:

Bsp.: CD 0:
CD

DIR - Befehl Zeige Inhaltsverzeichnis

Syntax: DIR (path)

gibt das Inhaltsverzeichnis des aktuellen oder angegebenen Pfades aus (entspricht dem CLI-Kommando DIR).

Bsp.: DIR 0: gibt Inhalt von Diskette im Laufwerk df0: aus

```
CD 1:C
DIR
```

gibt Inhalt des C-Verzeichnisses im Laufwerk df1: aus.

DIRA - Befehl Zeige Inhaltsverzeichnis von Verzeichnissen

Syntax: DIRA (path)

Bedienung wie DIR-Befehl, jedoch werden auch die Inhalte von Unterverzeichnissen ausgegeben (entspricht dem CLI-Kommando "dir (path) opt a").

MAKEDIR - Befehl Unterverzeichnis anlegen

Syntax: MAKEDIR (path)name

legt im Verzeichnis "path" oder im aktuellen ein Unterverzeichnis mit Namen "name" an. (entspricht dem CLI-Kommando makedir).

DELETE - Befehl Lösche Datei

Syntax: DELETE (path)name

löscht die Datei "name" im angegebenen oder aktuellen Verzeichnis (entspricht dem gleichnamigen CLI-Kommando).

FORMAT - Befehl Formatiere Diskette im AmigaDOS Format

Syntax: a) FORMAT (name)
b) FORMATV (name)
c) FORMATQ (name)

zu a)
formatiert die Diskette im aktuellen Laufwerk (CD-Befehl) im AmigaDOS-Format ohne die Formatierung zu überprüfen.

zu b)
wie a), überprüft jedoch die formatierte Diskette auf Fehler (entspricht CLI-Kommando format).

zu c)
löscht eine bereits formatierte Diskette wieder (besonders schnell, entspricht CLI-Kommando format QUICK)

In dieser Art formatierte Disketten können sowohl vom Modul als auch vom Amiga normal verwendet, aber nicht gebootet werden. Um Disketten bootfähig zu machen, benutzen Sie bitte den INSTALL-Befehl des Moduls (mit Virusschutz!)

Bsp.: FORMATV LEEREDISK
FORMATV "LEEREDISK"
FORMAT

INSTALL - Befehl Diskette installieren

Syntax: INSTALL (bootblocknr)

installiert einen Bootblock auf der Diskette im aktuellen Laufwerk (CD-Befehl). Als Bootblöcke stehen Ihnen zwei zur Auswahl: bootblocknr = 0 -> normaler Bootblock, wie er auch vom AmigaDOS verwendet wird, bootblocknr = 1 -> Virusprotector Bootblock, der verdächtige Programme anzeigt und beim Booten den Bildschirm hellgrün färbt, falls nichts verdächtiges gefunden wurde.
Da der Virusprotector mit den Kickstartversionen 1.2 und 1.3 ohne weiteres zusammenarbeitet, ist es sinnvoll stets diesen Bootblock (1) zu installieren, was einen zusätzlichen Schutz vor Viren garantiert (Bildschirm wird beim Booten nicht grün: Gefahr).

Bsp.: CD 1:
INSTALL 1

installiert den Virusprotector-Bootblock auf der Diskette im Laufwerk df1:

DISKCHECK - Befehl Diskette nach Fehlern untersuchen

Syntax: DISKCHECK (drive)

untersucht das angegebene Laufwerk (oder das aktuelle) nach Fehlern in der DOS-Struktur. Die Fehler werden tabellarisch aufgelistet.

Bsp.: DISKCHECK 2

Diskette im Laufwerk df2: überprüfen

DISKWIPE - Befehl Diskette schnell löschen

Syntax: DISKWIPE (drive)

löscht alle Daten auf einer Diskette im angegebenen Laufwerk

(oder aktuellem) besonders schnell und gründlich, indem jeder Track "zerstört" wird. So behandelte Disketten können erst nach einem Formatieren mit dem FORMAT-Befehl wieder verwendet werden.

Bsp.: DISKWIPE
DISKWIPE 0
DISKWIPE 3

TYPE - Befehl Datei anzeigen (ASCII)

Syntax: TYPE (path)name

stellt die Datei "name" im angegebenen oder aktuellen Verzeichnis als Text auf dem Bildschirm dar.

Bsp.: TYPE 0:S/STARTUP-SEQUENCE

Diskettenmonitor - Befehle

RT - Befehl Tracks lesen

Syntax: RT strack (num (dest))

liest vom aktuellen Laufwerk beginnend mit "strack" (0-!159) soviele Tracks ein, wie mit "num" angegeben (einen falls nichts weiter angegeben wurde). Die Daten werden dabei nacheinander ab der Adresse "dest" im Speicher des Amiga abgelegt.

Wird als Zieladresse "dest" nichts angegeben, so wird ein möglichst großer freier Bereich im Amiga Speicher gesucht. Dieser Bereich ist dann für spätere Diskettenzugriffe reserviert. Der Vorteil dieses Diskettenpuffers liegt darin, daß bei Verwendung desselben keine Daten im Speicher verlorengehen, also der Computer nach dem X-Befehl nicht nach Indien reist, sondern weiterhin seinen Dienst tut, als wäre nichts passiert.

Damit dies funktioniert, wird bei Verwendung eines Diskettenpuffers jedesmal beim Verlassen des Moduls der belegte Speicher wiederhergestellt. Das merkt man daran, daß bei Eingabe von X eine Sicherheitsabfrage erscheint, ob man das Modul verlassen will, ohne den Diskettenpuffer zurückzusetzen, was man im Normalfall mit N oder auch nur mit der RETURN-Taste quittiert. Lediglich für den Fall, daß die Diskettenpufferdaten nicht gelöscht werden sollen, kann man dies mit Y verhindern, was aber aus bekannter Ursache zu Systemabstürzen führen kann.

IM ZUSAMMENHANG MIT DIESEM DISKETTENPUFFER wurde ein neues Zahlenformat entwickelt, das einem das lästige Umrechnen vom Tracks, Sektoren und Offsets auf die tatsächliche Adresse im Speicher erleichtert. Und zwar kann bei allen Befehlen (insbesondere den Monitorbefehlen) anstelle von Adressen auch

Track, Sektor und Offset eingegeben werden:

Syntax: a) Ttrack(Ssector(Ooffset))
b) Ssector(Ooffset)

Beispiele für Diskettennamen (Rootblock Track !80 Sektor 0 Offset !433) gemäß Syntax a):

T!80S0O!433 oder T50S0O1B1 oder T!80S0O1B1 u.s.w.

gemäß Syntax b):

S!880O!433 oder S370O1B1 u.s.w.

Solche Ausdrücke werden vom Modul stets als Adressen interpretiert, und zwar werden die Adressen folgendermaßen gewonnen:

a) Adresse = Start des Diskpuffers + track*(!11*!512)
+ sector*!512
+ offset

b) Adresse = Start des Diskpuffers + sector*!512 + offset

Im folgenden illustrierenden Beispiel wird der Diskettenname der Disk im internen Laufwerk ausgegeben:

CD 0:

internes Laufwerk anwählen, falls noch nicht geschehen

RT !70 !20

lädt !20 Tracks beginnend mit dem Track !70 von der Disk im internen Laufwerk in den Diskkettenpuffer

N T!80S0O!433

berechnet automatisch die Adresse des Diskettennamens im Speicher und zeigt diesen an (siehe auch N-Befehl)

WT - Befehl Tracks schreiben

Syntax: WT strack num source

schreibt auf die Diskette im aktuellen Laufwerk beginnend ab dem Track "strack" (0-!159) "num" Tracks. Die Daten, die geschrieben werden sollen stehen hierbei ab der Adresse "source". Hiermit weisen wir nochmals darauf hin, daß anstelle jeder Adresse, also auch dieser, falls mit einen Diskettenpuffer gearbeitet wird (RT-Befehl ohne "dest"-Angabe) auch das spezielle "track/sektor/offset"-Format verwendet werden kann.

Bsp.: Einlesen und wieder Abspeichern von zwei Tracks (Track !10 und !11) mit und ohne Diskettenpuffer

RT !10 2

```
WT !10 2 T!10
```

das war mit Diskettenpuffer

```
RT A 2 1000
WT A 2 1000
```

ohne Diskettenpuffer, die Daten stehen jetzt "für immer" ab $1000.

DMON - Befehl Zeige Diskettenpuffer an

Syntax: DMON

gibt aus, an welcher Position im Speicher sich der gerade verwendete Diskettenpuffer befindet (nach RT-Befehl).

Bsp.: DMON

CLRDMON - Befehl Diskettenpuffer zurücksetzen

Syntax: CLRDMON

löscht sämtliche Daten im Diskettenpuffer, indem der alte Speicherinhalt wieder restauriert wird.

Bsp.: CLRDMON

BOOTCHK - Befehl Berechne BootBlock Checksum

Syntax: BOOTCHK address

berechnet die BootBlock Checksumme der beiden Sektoren, die sich ab der Adresse "adress" befinden und trägt sie an der richtigen Stelle ein.

Bsp.: RT 0
```
BOOTCHK T0S0
```

DATACHK -Befehl Berechne Datenchecksumme

Syntax: DATACHK address

berechnet die Datenblock Checksumme des Sektors, der sich ab der angegebenen Adresse befindet und trägt sie an der richtigen Stelle ein.

BAMCHK - Befehl Berechne Bitmapchecksumme

Syntax: BAMCHK address

dito mit einem BAM-Sektor

Maschinensprache-Monitor - Befehle

A - Befehl Direkt Lineassembler

Syntax: A address

ruft den Direkt-Assembler auf. Dabei wird zunächst die angegebene Adresse ausgegeben. Jetzt können Sie den gewünschten Maschinenbefehl in der üblichen Assembler-Schreibweise eingeben. Durch einfaches Drücken der ESC-Taste oder der RETURN-Taste ohne Eingabe eines Assembler-Befehls, kann der Direkt-Assembler wieder verlassen werden.

Bsp.: A 700000
;070000 ADDQ.L #1,D0
;070002 RTS
;070004

B - Befehl Breakpoint Setzen/Löschen

Syntax: a) B
b) BS address
c) BD address
d) BDA

zu a)
zeigt die derzeitig gesetzten Breakpoints an.

zu b)
setzt einen Breakpoint auf die angegebene Adresse. Falls kein Breakpoint mehr eingefügt werden kann, wird die Fehlermeldung "NO FREE ENTRY!" ausgegeben. Es sind dabei maximal 5 Breakpoints möglich!

zu c)
löscht den Breakpoint auf der angegebenen Adresse. Falls dieser nicht gesetzt war, wird die Fehlermeldung "BREAKPOINT NOT FOUND" ausgegeben.

zu d)
löscht alle gesetzten Breakpoints.

Die Breakpoints werden erst beim Verlassen des Moduls aktiviert, wobei Breakpoints, die auf den aktuellen PC gesetzt wurden, nicht aktiviert werden, da das Modul sonst sofort wieder aufgerufen werden würde.
Arbeitet das laufende Programm einen gesetzten Breakpoint ab, wird das Programm automatisch unterbrochen und der Modul-Editor aktiviert sich. Außerdem wird die Meldung "BREAK-POINT RAISED AT" und die Adresse des Breakpoints ausgegeben.

WAS IST EIN BREAKPOINT?

Ein Breakpoint (dt. Stoppunkt) ist ein Befehl. Wenn der Prozessor diesen Befehl abgearbeitet hat, wird das Programm unterbrochen und ein anderes Programm, in diesem Fall der Modul-Editor, aufgerufen.
Anwendung: Man ist in der Lage das laufende Programm gezielt an einer ganz bestimmten Stelle abzubrechen und mit dem Modul-Editor zu analysieren.

Bsp.: BS 4348
BREAKPOINT INSERTED
B
BREAKPOINTS: 004348

C - Befehl Copper Ass-/Disassembler

Syntax: C 1 ¦ 2 ¦ address

disassembliert die Copper-Liste ab der angegebenen Adresse. Falls anstelle der Adresse die Ziffer 1 oder 2 angegeben wurde, wird die Adresse der ersten bzw. zweiten aktuellen Copper-Liste verwendet. Das daraus resultierende Listing wird direkt vom Copper-Assembler verarbeitet, so daß Sie direkt in der Copperliste editieren können. (RETURN in jeder Zeile nicht vergessen!)

WAS IST DER COPPER?
Der Copper ist ein Amiga-Spezial-Chip welcher wie ein Prozessor sein eigenes Programm abarbeitet. Im Grunde genommen kann der Copper nur festgelegte Daten in die Chipregister schreiben (ab der Adresse $dff000). Da er dies aber an genau festgelegten Bildschirmstrahl-Positionen machen kann, lassen sich erstaunliche Effekte erzielen. Der Copper wird normalerweise für den Bildschirmaufbau verwendet.

WAS IST EINE COPPER-LISTE?
Die Copper-Liste ist nichts anderes, als ein Programm für den Copper.

Syntax der Copper-Befehle

MOVE #daten,adr
daten := zu schreibende Daten
adr := Offset des zu beschreibenden Chipregisters
WAIT (x,y,xmask,ymask,bfd)
x := vertikale Strahlposition
y := horizontale Strahlposition
xmask := vertikale Maske
ymask := horizontale Maske
bfd := Blitter-finished-disable-Bit
SKIP (x,y,xmask,ymask,bfd)
parameter siehe WAIT-Befehl

Bsp.: Ändern der Hintergrundfarbe der Workbench
C 2
~00004436 WAIT ($0,$2F,$1FC,$7F,$1)
~0000443A MOVE #$005A,$100
↑↑↑
ändern!

Nach dem Verlassen des Moduls wird man die geänderte Hintergrundfarbe bewundern können.

COMP - Befehl Vergleiche Speicherbereiche

Syntax: COMP start end dest

vergleicht den Speicherbereich von "start" bis "end" mit dem Speicherbereich ab "dest" und gibt Unterschiede anhand ihrer Adressen im "dest"-Bereich aus.

Bsp.: M 100
:000100 12 11 01 12 12 01 00 00 .....
COMP 100 103 103
000104
READY.

D - Befehl Disassembler

Syntax: D (0 ¦ address)

disassembliert ein 68000 Maschinenprogramm ab der angegebenen Adresse. Falls keine Adresse angegeben wurde, wird ab der zuletzt disassemblierten Zeile (voriger D-Befehl) weitergemacht. Sollte der erste vorkommende D-Befehl ohne Adresse eingegeben werden, wird als Adresse die genommen, bei der das Programm beim Verlassen des Moduls fortgesetzt wird. Gibt man anstelle der Adresse die Ziffer 0 an, wird für die Adresse der Program-Counter eingesetzt.

Bsp.: D 0
~01234 SUBQ.W #1,$000500

E - Befehl Editiere Chipregister

Syntax: E (registeroffset)

zeigt den Inhalt des mit "registeroffset" angegebenen Custom Chip-Registers hexadezimal und binär an. Wird kein Register angegeben, wird das Register 0 ausgegeben. Der Inhalt dieses Registers kann nun durch einfaches Überschreiben (plus RETURN) geändert werden. Der geänderte Inhalt wird jedoch nicht direkt, sondern erst beim Verlassen des Moduls gesetzt.

Wie Sie vielleicht schon bemerkt haben, stehen ja zwei gleiche Werte (Hexadezimal und Binär) hinter der Chipregisternummer. Wenn Sie nun das Chip-Register verändern wollen, dürfen Sie nur den Wert in EINEM der zwei Werte verändern z.B. den Hexadezimalen.

Man beachte, daß JEDES Chip-Register sowohl gelesen, als auch beschrieben werden kann. Da dies physikalisch nicht möglich

ist, wird dies vom E-Befehl logisch korrigiert und führt so dennoch in jedem Fall zu einem richtigen Ergebnis. Schreibt man zum Beispiel auf das Nur-Lese-Register "DMACONR" (Register $2), so wird dies automatisch auf das entsprechende Schreibregister umgeleitet, liest man das Nur-Schreibregister "COLOR00" aus (Hintergrundfarbe), so erhält man den zuletzt von dem PROZESSOR (!) in das Chip-Register geschriebenen Wert.
Oft kommt es vor, daß Werte, die man von Hand in die Chip-Register einträgt, bei jedem Bildaufbau (50 mal pro Sekunde) vom Copper überschrieben werden, so daß "von Hand" keine sichtbaren Wirkungen auftreten. So werden z.B. die Farben des aktuellen Bildes oft vom Copper in die jeweiligen Farbregister geschrieben, so daß Änderungen der Farbregister mit Hilfe des E-Befehls keine Wirkung zeigen, da Sie vom Copper ja wieder überschrieben werden! Wollen Sie in einem solchen Fall die Farben aber trotzdem ändern, müssen Sie das Copperprogramm (C-Befehl) entsprechend abändern.

Bsp.: E 9A
:09A 6302 %0110001100000010
↑
auf 0 setzen!
(dies sperrt sämtliche Interrupts)
nach Verlassen des Moduls (X-Befehl) sind sämtliche Interrupts gesperrt, d.h. zum Beispiel, daß sich der Mauszeiger nicht mehr rührt u.s.w.

Um nun aus diesem mißlichen Zustand wieder herauszukommen, drücken Sie den Modul-Taster und geben Sie folgende Befehle ein:
Bsp.: E 9A
:09A 4302 %0010001100000010
↑
auf 1 setzen!
(dies erlaubt wieder alle Interrupts!)

F - Befehl vielfältiger Such-Befehl

Syntax: a) F string(,start end)
b) FS string(,start end)
c) FR string(,start end)
d) FA address(,start end)
e) FAQ address(,start end)

zu a)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von "start" bis "end" nach dem angegebenen String und gibt die gefundenen Adressen der Reihe nach aus.

zu b)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von "start" bis "end" nach dem angegebenen String, macht aber keine Unterscheidung zwischen Groß- und Kleinbuchstaben (dadurch besonders geeignet ASCII-Texte zu finden)

zu c)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von "start" bis "end" nach dem angegebenen String, jedoch relativ, d.h. wird beispielsweise der String 0A 03 0B angegeben, sucht er nach folgender Bytefolge: xx xx-7 xx+1, wobei xx beliebig ist! So findet der FR-Befehl auch die Bytefolge 15 0E 16 oder 38 31 39. Außerdem wird nach der Adresse des gefundenen Strings noch der Offset ausgegeben, der die Differenz zwischen dem angegebenen und dem gefundenen String angibt. Der FR-Befehl findet so z.B. dann Anwendung, wenn nach einem Text gesucht werden soll, der in einem anderen Zeichencode, als dem ASCII-Code im Speicher steht, die Buchstaben aber dennoch, wie meist, in aufsteigender Reihe geordnet sind.

zu d)
durchsucht den gesamten Speicher oder, falls angegeben, den Speicherbereich von "start" bis "end" nach Maschinenbefehlen, die in irgendeiner Weise auf die angegebene Adresse zugreifen. Gefunden werden die Befehle, die mit folgenden Adressierungsarten auf die angegebene Adresse zugreifen:

- absolut kurz/lang
- Adressregister indirekt mit/ohne Adressdistanz
- PC-relative Adressierung mit Adressdistanz

zu e)
wirkt wie der FA-Befehl, nur ist der FAQ-Befehl ca. doppelt so schnell. Der FAQ-Befehl ist allerdings nicht so sicher wie der FA-Befehl, d.h. wenn der FAQ-Befehl nichts findet, so kann der FA-Befehl erfolgreicher sein!
ACHTUNG: Damit der FA/FAQ-Befehl alle gewünschten Zugriffe korrekt finden kann, muß der augenblicklich auf dem AMIGA laufende Task das zu untersuchende Programm sein (bei Spielen ist er das i.A. automatisch). Den Namen des augenblicklich laufenden Tasks erfährt man mit dem TASKS-Befehl (siehe dort).
Bsp.: F "ABC",100 5000

```
SEARCH FROM 000100 TO 005000
00000480
READY.
M 480
:00000480 41 42 43 2A 45 46 47 2A ABC*EFG*
FR "ABC"
SEARCH FROM 000000 TO 080000
0000480 OFFSET: 00
0000484 OFFSET: FC
READY.
FA 100 2200
SEARCH FROM 000000 TO 005000
00004024 ANDI.W #$1234,$00000100
READY.
```

G - Befehl Setze fort an Adresse

Syntax: G (address)

setzt das unterbrochene Programm an der angegebenen Adresse

fort. Wird keine Adresse angegeben, so wird der aktuelle PC verwendet, d.h. das Modul wird normal verlassen. Dieser Befehl sollte nur mit äußerster Vorsicht benutzt werden, da bei sinnlosen Adressen sich das unterbrochene Programm meist unwiederbringlich verabschiedet.

Bsp.: G FC00D2

springt in die Kickstart Reset-Routine -> es wird ein Soft-Reset ausgelöst.

LM - Befehl Lade Speicherbereich

Syntax: LM (path)name, dest

lädt den unter dem Namen "name" im Unterverzeichnis "path" oder dem aktuellen Unterverzeichnis abgespeicherten Speicherbereich (normale Datei) ab der angegebenen Adresse "dest" in den Speicher.

Bsp.: LM "MEM",100

M - Befehl Zeige und editiere Speicher

Syntax: M address

gibt den Speicher ab der angegebenen Adresse aus. Im ausgegebenen Listing kann dann direkt (nur im Hex-Bereich) editiert werden. (Nach geänderter Zeile RETURN nicht vergessen!)

Bsp.: M 234
:00000234 11 20 2a 00 10 ff fe fd . *.....

MEMCODE - Befehl Kodiert Speicherbereich (EOR-Verfahren)

Syntax: start end code

kodiert den Speicherbereich von "start" bis "end" mit der Kodezahl "code" (0-$ffffffff).

Der so kodierte Speicher kann wieder decodiert werden, indem man ihn nochmals mit demselben Kode zu kodieren versucht.

Bsp.: M 100
:00000100 40 41 42 43 44 45 46 47 ABCDEFGH
MEMCODE 100 108 1
M 100
:00000100 41 40 43 42 45 44 47 46 BADCFEHG
MEMCODE 100 108 1
M 100
:00000100 40 41 42 43 44 45 46 47 ABCDEFGH

N - Befehl Zeige Speicher als Text

Syntax: a) N address
b) NQ address
c) NO (offset)

zu a)
gibt den Speicher ab der angegebenen Adresse als Text aus. Der Text wird gewonnen, indem zu jedem Speicherbyte ein interner Offset (Konstante), der mit dem NO-Befehl vorgewählt werden kann, hinzuaddiert wird. Das Ergebnis wird dann als ASCII-Code interpretiert. Falls nichts anderes angegeben wurde, wird diese Konstante als 0 angenommen, d.h. normalerweise wird der Speicher direkt als ASCII-Text interpretiert. Es kann direkt im ASCII-Text editiert werden.

zu b)
gibt den Speicher ab der angegebenen Adresse als Text aus. Es werden aber alle nicht druckbaren Zeichen ausgelassen. Die Ausgabe kann wie gewöhnlich mit SHIFT gestoppt und mit ESC abgebrochen werden. Am Ende des NQ-Befehls wird die Adresse des zuletzt untersuchten Speicherbereichs ausgegeben.

zu c)
setzt den Offset beim N/NQ-Befehl auf den angegebenen Wert, oder gibt den aktuellen Offset aus. Hier kann z.B. auch der Wert Verwendung finden, der beim FR-Befehl als Offset ausgegeben wurde, um auch andere Texte als ASCII-kodierte lesen zu können.

Bsp.: M 1234
:001234 41 42 43 44 00 00 00 00 ABCD....
N 1234
;001234 ABCD.....
NO 1
N 1234
;001234 BCDE.....

O - Befehl Fülle Speicherbereich

Syntax: O string, start end

füllt den angegebenen Speicherbereich von "start" bis "end" mit dem angegebenen String. Falls der String kleiner ist wie der zu füllende Speicherbereich, wird er wiederholt in den Speicher geschrieben.

Bsp.: O "AMIGA",0 80000
N 0
; 000000 AMIGAAMIGAAMIGAAMIGAAMIGAAMIGAAM
O 0, 0 4
M 0
:000000 00 00 00 00 41 41 4D 49 ....AAMI

R - Befehl Zeige/editiere CPU Register

Syntax: R (register value)

setzt, falls angegeben, das CPU-Register auf den angegebenen Wert und gibt sämtliche Register auf dem Bildschirm aus. Für "register" schreibt man:

Datenregister: D0, D1, D2, ..., D7
Adressregister: A0, A1, ...A7
User Stackpointer: SP
Statusregister: SR
Programcounter: PC
Flags im SR: FT, FS, FV, FC, FZ oder FX
Interruptmaske: FI

Bsp.: R D0 1234

```
D0 = 00001234 D1 = 00000000
D2 = 11111111 D3 = 22222222
D4 = 22333333 D5 = 44444444
D6 = 77777777 D7 = 00000012
A0 = 00000000 A1 = 00000000
A2 = 00000000 A3 = 00000000
A4 = 00000000 A5 = 00000000
A6 = 00000000 A7 = 000014A6
PC = FC00D2   USP = 00123430 SR = 0010
T=0 S=0 I=000 X=0 N=1 Z=0 V=0 C=0
```

R PC FC00D2

setzt PC auf die Reset Routine des Kickstart

SETEXCEPT - Befehl Exceptionhandler installieren

Syntax: SETEXCEPT

installiert den Exception-Handler des Moduls, der bewirkt, daß bei den meisten Systemabstürzen (siehe unten) zunächst kein GURU ausgelöst wird, sondern das Modul aufgerufen wird. Es meldet sich dabei mit der Meldung, welche Exception den GURU verursacht hat und wo etwa der fehlerhafte Befehl steckt.
Abgefangen werden folgende Exceptions:

- Adressfehler
- nichtcodierter OP-Code
- Division durch Null
- Line A Emulator
- Line F Emulator

SM - Befehl Speicherbereich abspeichern

Syntax: a) SM (path)name, start end
b) SMDATA (path)name, start end
c) SMDC (path)name, start end

zu a)
speichert den Speicherbereich von "start" bis "end" unter dem Namen "name" im aktuellen oder angegebenen Verzeichnis auf Diskette ab. Beachten Sie, daß der zu speichernde Speicherbereich "am Stück" im Speicher (RAM/ROM) des Amigas liegen muß und eine Länge von ca 800KB (bei einer leeren Diskette) nicht überschreiten darf.
Das erzeugte File enthält danach den Speicher direkt als Bytefolge (Binärfile).

zu b)
gleiche Funktion wie SM-Befehl, jedoch wird ein File erzeugt, das den angegebenen Speicherbereich in Form von BASIC DATA-Zeilen enthält. So gespeicherte Speicherbereiche können dann einfach in Basic Programme eingebunden (Merge-Befehl im BASIC) werden.

zu c)
gleiche Funktion wie SMDATA-Befehl, jedoch enthält das erzeugte File DC-Zeilen, wie sie von jedem Assembler verstanden werden, und kann dadurch (mit Include-Anweisung des Assemblers) direkt in einem Assembler Source-File verwendet werden.

Bsp.: SM "MEM", 100 1000

TRANS - Befehl Kopiere Speicherbereich

Syntax: TRANS start end dest

kopiert den Speicherbereich von "start" bis "end" in den Speicher des Amigas ab der Adresse "dest".

Bsp.: TRANS 100 200 10000

kopiert $100-$1FF (=$100 Bytes) nach $1000-$10FF

W - Befehl Zeige/editiere CIA-Register

Syntax: W (register)

stellt den Inhalt des angegebenen Registers der beiden CIA's auf dem Bildschirm dar. Die Inhalte der Register können direkt im Listing verändert werden. Wird keine Registernummer angegeben, so wird das CIA Registerpaar $0 ausgegeben. Der erste binär dargestellte Wert ist stets der Inhalt des CIA-A, der zweite der Inhalt von CIA-B.
Es werden dabei mehr als die physikalisch vorhandenen 16 Register pro CIA ausgegeben. Die Werte ab "Register" 16 sind CIA-interne Zustände, die auf diese Weise vom Modul gespeichert werden und nicht verändert werden sollten!

WAS IST EIN CIA?
Bei den CIA's ((C)omplex (I)nterface (A)dapter) handelt es sich um zwei 8520 IC's. Die CIA's haben z.B. die Aufgabe einige Diskettensignale zu verwalten und besitzen unter anderem noch mehrere frei programmierbare Timer.

Bsp.: W
'00 %11111100 %11111111
↑
auf 1 setzen!

verläßt man nun das Modul (X-Befehl) geht die Power LED aus!

X - Befehl Unterbrochenes Programm fortsetzen

Syntax: X

verläßt den Modul-Editor und setzt das unterbrochene Programm an der Stelle fort, an der es unterbrochen wurde (entspricht dem G-Befehl ohne Adresse).
Achten Sie darauf, daß eventuell vom Programm benötigte Disketten, wie zum Zeitpunkt des Abbruchs oder des Freezens, wieder eingelegt sind.

Bsp.: X

Y - Befehl Zeige/editiere Speicherbereich binär

Syntax: a) Y address
b) YS (bytes)

zu a)
gibt den Speicher ab der angegebenen Adresse binär aus und zwar soviele Bytes, wie mit Hilfe des YS-Befehls eingestellt worden sind.
Es kann direkt im Listing editiert werden. (RETURN nicht vergessen!)

zu b)
setzt die Byteanzahl "bytes" für den Y-Befehl auf den angegebenen Wert. Es können Werte zwischen 1 und 8 angegeben werden.
Wird keine Byteanzahl angegeben, so wird der augenblicklich eingestellte Wert ausgegeben. Es werden allerdings nicht die Bytes ausgegeben, sondern die Anzahl der Bytes in Bits (1 Byte = 8 Bit)

Bsp.: Y 100
.000100 %0011001100110011
YS 1
Y 100
.000100 %00110011
.000101 %00110011
YS
CURRENT BIT WIDTH: !08

? - Befehl Mini-Taschenrechner

Syntax: ? (-)value (+ ¦ - ¦ * ¦ / value) ...

berechnet den angegebenen Term OHNE Berücksichtigung der "Punkt vor Strich"-Regel und gibt das Ergebnis Binär, Dezimal und Hexadezimal aus.

Bsp.: ? !10+4*A
%00000000000000000000000000110010 = !50 = $32

Befehle um Betriebssystem-Parameter zu untersuchen

Naturgemäß funktionieren alle folgenden Befehle mit Ausnahme von INFO und EXCEPTIONS nur dann richtig, wenn das unterbrochene Programm das Betriebssystem auch nutzt, was bei Spielen nicht immer der Fall ist. Sollte das Betriebssystem nicht benutzt werden, d.h. sind die Betriebssystem Parameter erkennbar zerstört, wird die Meldung "EXECBASE NOT VALID" ausgegeben.

INFO - Befehl Zeige Amigastatus und einige Chipregs

Syntax: INFO (picnr)

stellt die wichtigsten System-Parameter des Amigas, die auch den Bildaufbau wesentlich bestimmen, dar. Bei manchen Bildern kommt es vor, daß das Bild sich aus mehreren Bildern zusammensetzt, ähnlich sich überlappenden Workbenchscreens. Die Systemparameter können sich nur auf einzelne Bilder beziehen. Welches Bild man nun meint, gibt man mit "picnr" an. Läßt man die Bildnummer weg, wird das erste Bild analysiert. U.a. werden die Farben und Adressen der Bitplane des aktuellen Screens angezeigt, aber auch Sprite- und Audiodaten und die aktuellen Positionen der Floppy-Schreib/Lese-Köpfe. Um deren Position zu bestimmen, muß übrigens der Lesekopf jedes Laufwerks bewegt werden, was man deutlich hören kann, einer eventuell eingelegten Diskette aber auf keinen Fall schadet!

Die Ausgabe des INFO-Befehls:
- einige Custom-Chip Register deren Bedeutung in der Fachliteratur nachzulesen ist.
- TDF0, TDF1, TDF2, TDF3 geben die aktuelle Lesekopfposition in Tracks an. "****" bedeutet, daß dieses Laufwerk nicht vorhanden ist.

DEVICES - Befehl Devices des Systems ausgeben

Syntax: DEVICES

INTERRUPTS - Befehl Interrupts des Systems ausgeben

Syntax: INTERRUPTS

Ausgegeben werden nur ordentlich dem Betriebssystem "mitgeteilte" Interrupts.

Um die Adressen der tatsächlichen Interruptroutinen zu erfahren, müssen Sie den EXCEPTIONS-Befehl verwenden.

LIBRARIES - Befehl System Libraries ausgeben

Syntax: LIBRARIES

PORTS - Befehl Ports des Systems ausgeben

Syntax: PORTS

RESOURCES - Befehl Resources des Systems ausgeben

Syntax: RESOURCES

TASKS - Befehl Ausgabe der laufenden Tasks

Syntax: TASKS

die Tasks werden dabei in drei Kategorien eingeteilt:
Running, Ready und Waiting Task.
Das Programm, das im Moment des Programmabbruchs bearbeitet wurde, steht unter dem Eintrag Running Task. Die Adressen vor den Tasks geben die Basisadresse der zugehörigen Task-Struktur an (siehe weiterführende Literatur)

AVAIL - Befehl Ausgabe des momentan noch freien Speichers

Syntax: AVAIL

CHIPREGS - Befehl Ausgabe der Chip-Register Namen

Syntax: CHIPREGS

gibt eine Liste der im Amiga verwendeten Custom-Chip-Register mit Namen und Offset aus.

EXCEPTIONS - Befehl Ausgabe Exceptions/Interrupts

Syntax: EXCEPTIONS

gibt die augenblickliche Belegung folgender (Prozessor-)Sprungvektoren aus:
Exceptions, Interrupt, Traps

EXECBASE - Befehl Execbase ausgeben

Syntax: EXECBASE

gibt Namen, Offsets und augenblickliche Inhalte sämtlicher ExecBase-Einträge aus.

Verschiedene Befehle

RAMTEST - Befehl Teste Speicherbausteine des Amiga

Syntax: RAMTEST start end

testet den angegebenen Speicherbereich von "start" bis "end" (muß am Stück liegen) und gibt fehlerhafte Bytes aus.

PACK - Befehl Komprimiere Speicherbereich

Syntax: PACK start end dest crate

komprimiert den Speicherbereich von "start" bis "end" und schreibt den komprimierten Speicher ab der Adresse "dest" wieder in den Speicher.
Mit "crate" (0-$7fff) wird die Effizienz des Komprimiervorgangs festgelegt. Normale Werte bewegen sich um $20 herum. Je größer "crate", desto besser wird komprimiert (dauert aber i.A. länger).
Nach dem Komprimiervorgang wird die Länge der komprimierten Daten ausgegeben. (Zur Verwendung beim Abspeichern und Entkomprimieren).

Bsp.: PACK 70000 70100 71000 30
000044
packt den Bereich von $70000-$700ff nach $71000
UNPACK 60000 71044
entpackt den oben komprimierten Bereich wieder, so daß jetzt von $60000 bis $60100 das Gleiche steht wie ab $70000.

UNPACK - Befehl Komprimierte Speicherbereiche entfalten

Syntax: UNPACK dest endofpacked

macht den Vorgang des PACK-Befehls wieder rückgängig, indem der komprimierte Speicherbereich wieder entkomprimiert nach "dest" geschrieben wird. Endofpacked muß die Adresse des letzten gepackten Bytes +1 enthalten.

Bsp.: s. PACK-Befehl

COLOR - Befehl Modulfarben setzen

Syntax: COLOR back pen

setzt die Farben des Moduleditors auf die gewünschten Werte, dabei ist "back" die Farbe des Hintergrundes und "pen" die der Zeichen.
Die eingegebenen Zahlen "back" und "pen" werden dabei als RGB-Farbwerte interpretiert (z.B. 000 = schwarz, F00 = rot, FFF = weiss u.s.w.)

Bsp.: COLOR 000 0F0
COLOR 05A FFF

RCOLOR - Befehl Modulfarben wieder auf Standardwerte setzen

Syntax: RCOLOR

TM - Befehl Editiere Speichermerker

Syntax: a) TM
b) TMS address
c) TMD address

zu a)
zeigt die aktuell gesetzten Speichermerker an

zu b)
setzt einen Speichermerker auf die nachfolgend eingegebenen Werte. Es sind maximal zehn Speichermerker möglich. Die Speichermerker stellen dabei einen Notizblock dar, auf dem alle möglichen in einem Spiel wichtigen Speicherstellen vermerkt werden können, die Sie vielleicht mit dem Trainermaker oder anders herausbekommen haben.
Die Speichermerker werden beim Speichern des Spiels mit dem SA-Befehl mit abgespeichert und automatisch beim Laden mittels LA/LR-Befehl wieder eingeladen!

Der TMS-Befehl verlangt folgende Eingaben:

TYPE	= Typ des Zählers - geben Sie hier den am nächsten passenden Typ an
COUNTERLENGTH	= Geben Sie nun die Länge des Zählers im Speicher an. 1 = Byte, 2 = Wort, 3 = Langwort (i.A. ist es ein Wort)
MAXCOUNT	= Geben Sie hier den größten Wert des Zählers an, bei dem das Programm noch korrekt arbeitet (!127 = $7F hat sich bewährt, da größte vorzeichenbehaftete Zahl)
PLAYER	= Geben Sie nun die Spielernummer an, bei dem diese Parameter gelten (normalerweise Spieler Nummer eins)

zu c)
löscht den Speichermerker, der auf die angegebene Adresse zeigt. Falls dieser nicht existiert, wird die Meldung "MEMORYCOUNTER NOT FOUND" ausgegeben.

SPR - Befehl Editiere Hardware-Sprites

Syntax: SPR nr1 ¦ address1 (nr2 ¦ address2)

Gibt den Sprite der Nummer "nr1" oder den Sprite, der ab der Adresse "adress1" im Speicher (Chip-RAM!) liegt, als editierbares Listing auf dem Bildschirm aus, wobei die Spritefarben durch die Ziffern 0 bis 3 dargestellt werden.
Wird eine zweite Spritenummer oder -adresse angegeben, wird der Sprite als "attached"-Sprite ausgegeben und seine Farben durch die Ziffern 0 bis 9 und A bis F dargestellt.
In jedem Fall stellt die Farbe 0 die durchsichtige Farbe dar .

Bsp.: von der Workbench aus:

```
SPR 0
~001480 1111110000000000
~001482 1222221000000000
~001484 1333321000000000
~001486 1333210000000000
~001488 1333321000000000
~00148A 1331332100000000
~00148C 0110133210000000
~00148E 0000013321000000
~001490 0000001332100000
~001492 0000000131000000
~001494 0000000010000000
~001496 0000000000000000
```

Jetzt kann der Mauszeiger beliebig verändert werden!
(Nach jeder geänderten Zeile RETURN nicht vergessen!)

ANHANG A

Handhabung des Trainers

Das AMIGA ACTION REPLAY CARTRIDGE stellt Ihnen mit dem T-Befehl einen leistungsfähigen 3-Pass-Trainer zur Verfügung, der in der Lage ist, nahezu jedes Spiel zu trainen.

Aber VORSICHT: Die meisten Spiele werden, wenn einmal getraint, schnell langweilig.

Um ein Spiel zu trainen, d.h. Sie mit beliebig vielen Bildschirmleben auszustatten, sind folgende Arbeitsschritte vonnöten:
1.) Das zu trainende Spiel normal einladen und starten
2.) Dem gestarteten Spiel die Anzahl der Bildschirmleben (kurz Leben) entnehmen (z.B. fünf)
3.) Die Cartridge aktivieren (Taster)
4.) Trainer-Mode starten mit dem TS-Befehl (im Bsp.: TS !5)
5.) Das Modul verlassen (X-Befehl) und solange spielen bis

sich die Anzahl der Leben verändert hat. Am besten hat sich in der Praxis die sogenannte Kamikazemethode bewährt: Sämtliche Gegner mitnehmen ohne auch nur einen einzigen Schuß abzufeuern...

6.) Dem Spiel die Anzahl der jetzt noch zur Verfügung stehenden Leben entnehmen (z.B. vier) und die Cartridge aktivieren (Taster)

7.) Den Trainvorgang fortsetzen mit dem T-Befehl (im Bsp.: T !4)

8a) Falls keine oder sehr viele Adressen ausgegeben werden bei Schritt 5) weitermachen!

8b) Merken Sie sich die wenigen Adressen und fangen Sie noch einmal bei Punkt fünf an. Wenn Sie Adressen bemerken, welche immer wieder vorkommen, so haben Sie ziemlich sicher die gesuchte Adresse gefunden. Verfahren Sie wie in Punkt (8c), nur müssen Sie nun ausprobieren, welche der gefundenen Adressen Sie verändern müssen.

8c) Falls genau eine Adresse übrigbleibt und der gesamte Speicher durchsucht worden ist, so haben Sie mit ziemlicher Sicherheit die Speicherstelle gefunden, mit deren Hilfe das laufende Programm die Leben mitzählt. Falls der Speicher noch nicht ganz durchsucht worden sein sollte, kann es dennoch sein, daß die gefundene Adresse bereits die gesuchte ist. Zur Sicherheit aber sollten Sie ab Punkt 5) nochmal wiederholen und überprüfen, ob die Adresse bestehenbleibt. Dann können Sie sicher sein, die richtige Adresse erwischt zu haben.

Wenn Sie mit dieser Prozedur eine verdächtige Speicherstelle ausfindig gemacht haben, können Sie diese erhöhen, um mehr Bildschirmleben zu erhalten (M-Befehl)

z.B.: Die gefundene Adresse lautet 23686. Geben Sie nun "M 23686" ein. Nun werden sechzehn Bytes ab der Adresse 23686 angezeigt. Der Cursor steht nun direkt auf dem Zähler und Sie können nun eine zweistellige Hexadezimalzahl eintragen. Geben Sie nun RETURN + ESC + X + RETURN ein. Das Modul wird nun verlassen und Sie können sich nun von der Leistung des Trainers überzeugen!

Sie können die Adresse bei den Speichermerkern eintragen (TM-Befehl) und so zu jedem Programm immer direkt, ohne erneut suchen zu müssen, die jeweilig wichtigen Adressen zur Verfügung zu haben.

Aber Vorsicht: nicht jedes Programm "verkraftet" beliebig hohe Werte für Ihre Leben. Der Wert $7F = !127 hat sich in der Praxis jedoch als relativ absturzsicher bewährt.

Noch eleganter, als den Zähler von Hand ständig zu erhöhen, ist die Methode, den Maschinenbefehl, der die jeweilige Speicherstelle erniedrigt, zu entfernen. Zu diesem Zweck bietet die cartridge den TFD-Befehl an. Dieser sucht (und entfernt) wenn möglich diese Befehle und macht Sie damit absolut unsterblich (am Bildschirm).

Assembler Profis können mit dem FA/FAQ-Befehl sich genau informieren, wie, wann und wo auf den Zähler zugegriffen wird!

Auf gleiche Weise kann man alles trainen, was irgendwie abzählbar ist. Z.B.: Sie haben zu Beginn des Spiels drei

Hyper-Bomben zur Verfügung. starten Sie mit "TS !3". Jetzt werfen Sie eine Bombe und fahren mit "T ! 2" fort usw. Auf diese Weise finden Sie die Speicherstelle, mit der das Programm die Hyperbomben zählt.

Sollten Sie während des Trainvorgangs so viele Leben verlieren, daß das Spiel endet (GAME OVER): Macht nichts! Starten Sie einfach das Spiel erneut und setzten den Trainvorgang wie gewohnt fort, als wäre nichts geschehen. Lediglich die Anzahl der Leben hat sich ja erhöht.

ACHTUNG: Falls der Trainer bzw. der Final Train Befehl (TFD) nicht zum erwünschten Ergebnis führen sollte, lesen Sie sich bitte das Kapitel TIPS&TRICKS durch und probieren Sie die dort genannten Variationen aus! Mit etwas Übung können Sie dann fast jedes Programm trainen.

ANHANG B

Tips & Tricks

– Schalten Sie, falls möglich, Ihre Speichererweiterungen mit der Preferences-Seite ab. Sie ersparen sich dann unnötig lange Speicher- und Ladezeiten und unnötigen Diskettenbedarf beim SA/LA/LR-Befehl. Die Zeiten für die Suchbefehle und nicht zuletzt den Trainer werden ebenfalls verkürzt.

– Sollten Sie mit resetfesten Programmen arbeiten, müssen Sie das Speicher löschen und den Virustest in der Preferences-Seite abschalten.

– Wenn die Speicherlöschoption eingeschaltet ist, arbeitet der Packer beim Freezen besser und schneller.

– Sollten Sie über eine andere Speichererweiterung als eine interne Speichererweiterung um max 1.8 MByte verfügen, müssen Sie, falls Sie ein mit dieser Speichererweiterung gefreeztes Programm abspeichern wollen, Ihre Speichererweiterung von Hand mit Hilfe des SM-Befehls (eventuell nach packen mit PACK-Befehl) abspeichern, da das Modul sonst nur die 512k Chipmem abspeichert. Beim Einladen des gefreezten Programms müssen Sie dann wieder umgekehrt Ihre von Hand abgespeicherte Speichererweiterung mit dem LM-Befehl wieder einladen (und evtl. entpacken mit UNPACK-Befehl).

– Es ist ratsam, wegen der grassierenden Virusgefahr den Resetvirustest und das Speicherlöschen beim Reset eingeschaltet zu lassen, so daß sich keine Viren in Ihrem Amiga festsetzen können. Sollte bei einem Reset ein Virus erkannt werden (am Flackern des Bildschirms beim Reset zu erkennen), sollten Sie Ihre seit dem letzten Reset benutzten Disketten auf Viren überprüfen!

– Sollte der Trainer einmal versagen, nicht verzagen:
Geben Sie beim Trainvorgang (TS/T-Befehl) jeweils die Anzahl der Bildschirmleben (o.a.) plus eins (!) an.

Achten Sie darauf, daß sie den Zahlenwert immer korrekt angegeben haben. (Immer "!" für dezimale Zahlen eingegeben?)

– Bei manchen Zählern kann es vorkommen, daß der Zählerwert im BCD-Zahlenformat im Speicher steht. z.B.:
Auf dem Bildschirm steht der Zählerwert "978". Normalerweise müßten Sie jetzt mit "TS !978" den Trainvorgang starten. Wegen des BCD-Zahlenformates müssen Sie aber stattdessen bei T/TS-Befehl das "!" für dezimal weglassen. Der richtige Befehl lautet also "TS 978"!

– Sollte der TF-Befehl einmal nicht fündig werden und die Adresse der Speicherstelle ist ungerade (Endziffer 1, 3, 5, 7, 9, B, D, F), sollten Sie einmal von der Adresse eins abziehen und TF/TFD-Befehl wiederholen!
z.B.: Der Trainer liefert die Adresse $33EF für den Zähler. Der Befehl "TFD 33EF" hatte aber nichts gefunden, dann probieren Sie einmal die Zähleraaresse-1 (im Beispiel "TFD 33EE").

– Bei Zählern, die größer als !255 werden können (z.B. Score), liefert der Trainvorgang, wie auch sonst, die Adresse des letzten Bytes, d.h. wenn man das oder die vorhergehenden Bytes verändert, können evtl. sehr große Zählerwerte erreicht werden. Das Byte mit der niedrigsten Adresse dieses Zählers sollte jedoch nicht größer als $7F gewählt werden.

– Den zweiten Bildschirm (erreichbar über F10, zurückschalten über F10) nutzt man am besten für Notizen, da er nach Verlassen des Moduls nicht gelöscht wird.

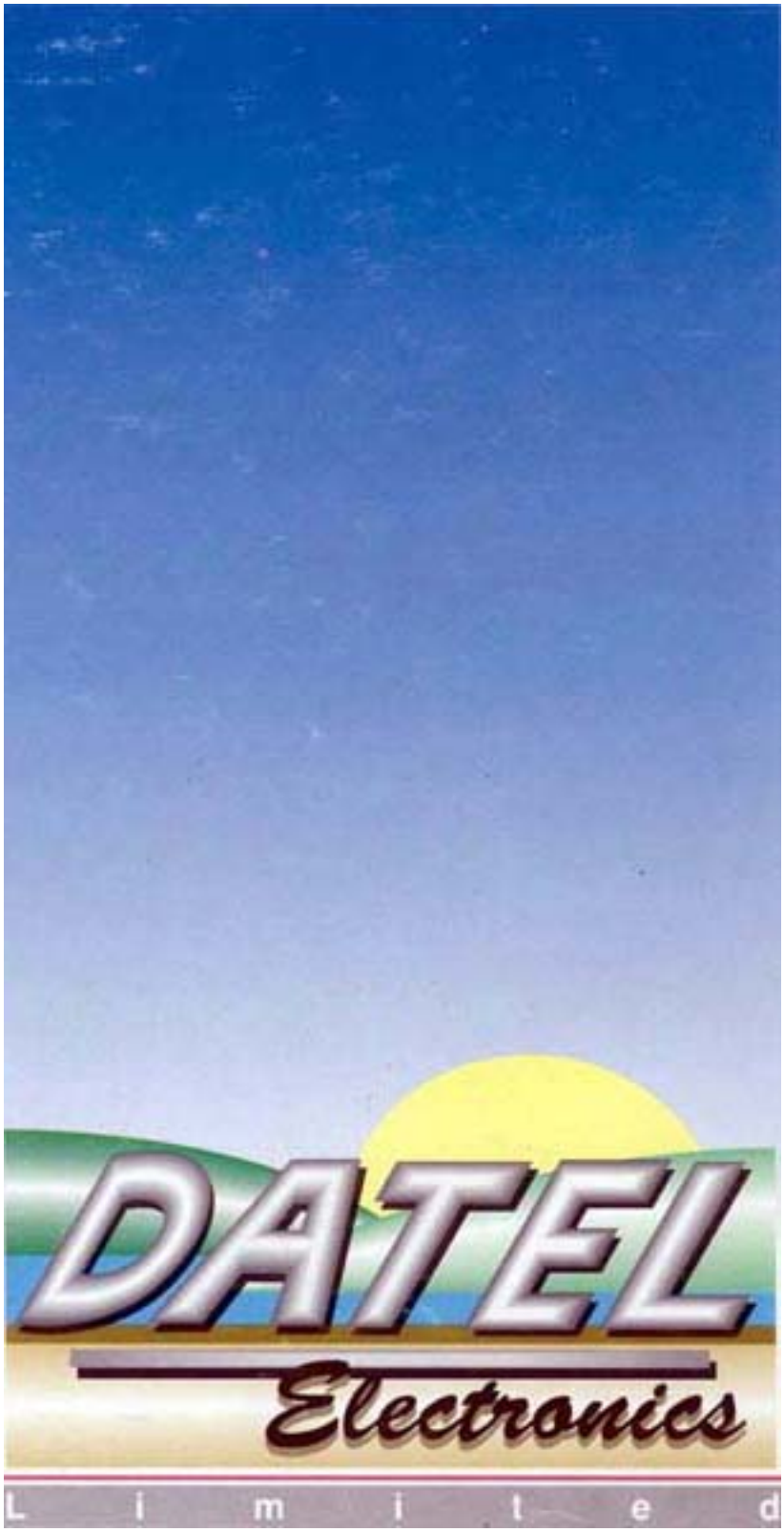
– Normalerweise sind alle Playerdaten wie z.B. Leben, Energie und Punkte sehr nahe im Speicher (RAM) aneinander angeordnet. Wenn Sie also z.B. den SCORE gefunden haben, müssen Sie nur in diesem Speicherbereich suchen und mit ziemlicher Sicherheit werden Sie nun auch die anderen Playerdaten finden!

– Wie mache ich mir eine Arbeitskopie von einem kopiergeschützten Original?
1) Falls es kein Nachladeprogramm ist, laden Sie einfach das Original ein und FREEZEN Sie es einfach mit dem SA-Befehl ->Fertig!!!
2) Es handelt sich um ein Original, welches Daten von der Diskette während dew Spiel nachlädt.
– Kopieren Sie Ihr Original mit einem üblichen Kopierprogramm.
– Laden Sie das Original ein.
– Nachdem der Kopierschutz abgefragt wurde, FREEZEN Sie es einfach (SA-Befehl) -> fertig.

– Wie lade ich meine Arbeitskopie meines Originals?
1) Falls es kein Nachladeprogramm ist, laden Sie einfach das gefreezte Programm mit dem LR-Befehl ein -> fertig.
2) Es handelt sich um eine Arbeitskopie, welche von Diskette nachlädt:
– Laden Sie das gefreezte Programm mit dem LA-Befehl ein.
– Legen Sie, ohne das Modul zu verlassen, die Kopie(n) der Originaldiskette(n) in Ihren Laufwerken.
– Verlassen Sie das Modul mit dem X-Befehl -> fertig.

– Wenn sie mit dem TF/TFD/FA/FAQ-Befehl arbeiten wollen, sollten Sie darauf achten, daß der gerade laufende Task (siehe TASKS-Befehl) das Spiel ist!
z.B.: Wollen Sie in einem Spiel die Adresse finden, welche ihren Zähler schändlich behandelt (ihn vermindert), so sollten Sie darauf achten, daß der aktuelle Task (TASKS-Befehl) auch das Spiel ist (sofern das Betriebssystem des Amiga nicht sowieso vom Spiel umgangen wird). Mit etwas Erfahrung werden Sie später schnell erkennen, ob der aktuelle Task der gesuchte Task ist.

– Falls ein Programm nicht mit dem Action Replay Mk II Modul arbeitet, sollten Sie die erweiterten Funktionen des Action Replay abschalten z.B: Virustest, Dauerfeuer, Bootselector, Diskcoder, Bootblockcoder, Speicherlöschen.



GOVAN ROAD, FENTON INDUSTRIAL ESTATE
FENTON, STROKE-ON-TRENT, ST4 2RS, ENGLAND.
TEL 0782 744707. FAX 0782 744292.

REFERENZKARTE

COMMANDS FOR SYSTEM INFORMATION:

INTERRUPTS:	SHOW EXECBASE INTERRUPTLISTS	- INTERRUPTS
EXCEPTIONS:	SHOW EXCEPTION- AND INTERRUPTVECTORS	- EXCEPTIONS
EXECBASE:	SHOW WHOLE EXECBASESTRUCTURE	- EXECBASE
AVAIL:	SHOW FREE MEMORY	- AVAIL
INFO:	SHOW IMPORTANT SYSTEMPARAMETERS	- INFO
LIBRARIES:	SHOW EXECBASE LIBRARYLIST	- LIBRARIES
RESOURCES:	SHOW EXECBASE RESOURCELIST	- RESOURCES
CHIPREGS:	SHOW NAME + OFFSET OF CHIPREGISTERS	- CHIPREGS
DEVICES:	SHOW EXECBASE DEVICELIST	- DEVICES
TASKS:	SHOW EXECBASE TASKLISTS	- TASKS
PORTS:	SHOW EXECBASE PORTLIST	- PORTS

DISK AND DISKCODING COMMANDS

BOOTCODE:	SHOW/SET BOOTBLOCK CODENUMBER	- BOOTCODE (CODENUMBER)
BOOTPROT:	CODE BOOTBLOCK OF ACTIVE DRIVE	- BOOTPROT (CODENUMBER)
CODE:	SHOW/SET DISK CODENUMBERS	- CODE (DRIVE CODENUMBER)
DCOPY:	BACKUP AMIGADOS DISKS	- DCOPY SOURCE DEST
CODECOPY:	DISKCOPY + DECODE SOURCE + CODE DEST	- CODECOPY SOURCE DEST
CD:	SHOW/CHANGE CURRENT MODULE-PATH	- CD (PATH)
DIR:	SHOW DISK-DIRECTORY	- DIR (PATH)
DIRA:	SHOW WHOLE DISK-DIRECTORY	- DIRA (PATH)
MAKEDIR:	CREATE DIRECTORY	- MAKEDIR PATH
DELETE:	DELETE FILE	- DELETE (PATH)FILENAME
FORMAT:	FORMAT DISK IN ACTIVE DRIVE	- FORMAT (NAME)
FORMATQ:	FORMAT DISK QUICK	- FORMATQ (NAME)
FORMATV:	FORMAT DISK AND VERIFY FORMAT	- FORMATV (NAME)
INSTALL:	INSTALL DISK IN ACTIVE DRIVE	- INSTALL (BOOTBLOCKNR.)
DISKCHECK:	CHECKS DISK FOR ERRORS	- DISKCHECK (DRIVE)
DISKWIPE:	CLEARS A DISK VERY FAST	- DISKWIPE (DRIVE)
TYPE:	TYPE FILE ON SCREEN	- TYPE (PATH)FILENAME

FREEZER AND RIPPER COMMANDS:

SA:	SAVE CURRENT PROGRAM TO DISK	- SA (PATH)NAME(,CRATE)
SR:	SAVE CURRENT PROGRAM AND START	- SR (PATH)NAME(,CRATE)
LA:	LOAD FREEZEFILE FROM DISK	- LA (PATH)NAME
LR:	LOAD FREEZEFILE FROM DISK AND START	- LR (PATH)NAME
SLOADER:	SAVE LOADER TO ACTIVE DRIVE	- SLOADER
LQ:	LOAD ALL FROM RAMDISK	- LQ
LQR:	LOAD ALL FROM RAMDISK AND RESTART	- LQR
SQ:	SAVE ALL TO RAMDISK	- SQ
SQR:	SAVE ALL TO RAMDISK AND RESTART	- SQR
TRACKER:	RIPS SOUNDTRACKER-MODULS IN MEMORY	- TRACKER START
SCAN:	SCAN MEMORY FOR SAMPLES	- SCAN
SP:	SAVE CURRENT PICTURE TO DISK	- SP (PATH)NAME(,NR HEIGHT)
P:	SHOW CURRENT PICTURE/MEMPEEKER	- P (PICNR)
SPM:	SAVE PICTURE OF MEMPEEKER	- SPM NAME

DISKMONITOR COMMANDS:

RT:	READ TRACKS FROM ACTIVE DRIVE	- RT STRACK (NUM DEST)
WT:	WRITE TRACKS TO ACTIVE DRIVE	- WT STRACK NUM SOURCE
DMON:	DISPLAY RANGE OF DISK-MON BUFFER	- DMON

CLRDMON:	RESTORE DISK-MON BUFFER	- CLRDMON
BOOTCHK:	SET CORRECT BOOTBLOCKCHECKSUM	- BOOTCHK SECTORADDR.
DATACHK:	SET CORRECT DATACHECKSUM	- DATACHK SECTORADDR.
BAMCHK:	SET CORRECT BITMAPCHECKSUM	- BAMCHK SECTORADDR.

TRAINER COMMANDS:

TS:	START TRAINER/TRAINERMODE	- TS STARTLIVES
T:	CONTINUE TRAINER	- T AKTLIVES
TX:	EXIT TRAINERMODE	- TX
TF:	SEARCH FOR DECREMENTING OPCODES	- TF ADDRESS
TFD:	SEARCH AND REMOVE DECREMENT OPCODES	- TFD ADDRESS
PC:	SHOW CURRENT PICTURE + ENERGY COUNT	- PC (PICNR)

MISC. COMMANDS

RAMTEST:	CHECKS MEMORYBLOCK FOR HARDERRORS	- RAMTEST START END
PACK:	PACKS MEMORY	- PACK START END DEST CRRATE
UNPACK:	UNPACKS WITH PACK-COMMAND PACKED MEM-	UNPACK DEST END OF PACKED
COLOR:	SET MODUL-EDITOR COLORS	- COLOR BACK PEN
RCOLOR:	RESET MODUL-EDITOR COLORS	- RCOLOR
TM:	SHOW REMARKS ABOUT CURR. PROGRAM	- TM
TMS:	SET REMARK ABOUT CURR. PROGRAMADDR.	- TMS ADDR
TMD:	DELETE REMARK ABOUT PROGRAM	- TMD ADDR
SPR:	SHOW/EDIT SPRITES	- SPR NR¦ADDR (NR¦ADDR)

VIRUS COMMANDS:

VIRUS:	SEARCH VIRUS IN MEMORY	- VIRUS
KILLVIRUS:	SEARCH AND REMOVE VIRUS IN MEMORY	- KILLVIRUS

MONITOR COMMANDS:

SETEXCEPT:	SET EXCEPTION HANDLER (NO MORE GURU)	- SETEXCEPT
COMP:	COMPARE MEMORYBLOCKS	- COMP START END DEST
LM:	LOAD FILE TO MEMORY	- LM (PATH)NAME, DEST
SM:	SAVE MEMORY BLOCK TO DISK	- SM (PATH)NAME, START END
SMDC:	SAVE MEMORY BLOCK TO DISK AS DC.B	- SMDC (PATH)NAME, START END
SMDATA:	SAVE MEMORY BLOCK TO DISK AS DATA	- SMDATA (PATH)NAME, START END
A:	START M68000 ASSEMBLER	- A ADDRESS
B:	SHOW CURRENT BREAKPOINTS	- B ADDRESS
BS:	SET BREAKPOINT	- BS ADDRESS
BD:	DELETE BREAKPOINT	- BD ADDRESS
BDA:	DELETE ALL BREAKPOINTS	- BDA
X:	RESTART CURRENT PROGRAM	- X
C:	COPPERASSEMBLER/DISASSEMBLER	- C 1¦2¦ADDRESS
D:	M68000 DISASSEMBLER	- D (0¦ADDRESS)
E:	SHOW/EDIT CHIPREGISTERS	- E (OFFSET)
F:	SEARCH FOR STRING (CASESENSITIVE)	- F STRING(,START END)
FA:	SEARCH FOR ADR ADDRESSING OPCODE	- FA ADDRESS (START END)
FAQ:	FASTSEARCH FOR ADR ADDRESSING OPCODE	- FAQ ADR (START END)
FR:	SEARCH FOR RELATIVE-STRING	- FR STRING(,START END)
FS:	SEARCH STRING (NOT CASESENSITIVE)	- FS STRING(,START END)
G:	RESTART PROGRAM AT ADDRESS	- G ADDRESS
TRANS:	COPY MEMORYBLOCK	- TRANS START END DEST
WS:	WRITE STRING TO MEMORY	- WS STRING, ADDRESS
M:	SHOW/EDIT MEMORY AS BYTES	- M ADDRESS
MEMCODE:	CODES MEMORY (EOR.B)	- MEMCODE START END CODE
N:	SHOW/EDIT MEMORY AS ASCII	- N ADDRESS
NO:	SHOW/SET ASCII-DUMP OFFSET	- NO (OFFSET)

NQ:	DISPLAY MEMORY QUICK AS ASCII	- NQ ADDRESS
O:	FILL MEMORYBLOCK WITH STRING	- O STRING, START END
R:	SHOW/EDIT PROCESSOR REGISTERS	- R (REG VALUE)
W:	SHOW/EDIT CIA'S	- W (REGISTER)
Y:	SHOW/EDIT MEMORY AS BINARY	- Y ADDRESS
YS:	SET DATAWIDTH IN Y COMMAND	- Y BYTES
?:	CALCULATOR	- ? (+ ¦ - ¦ * ¦ / VALUE)

NUMBER FORMATS:

HEXADECIMAL: $12AB OR 12AB
DECIMAL: !15, !880
BINARY: %001110101, %101
DISKMONITOR (IF USING RT AND WT COMMANDS WITH DISKBUFFER):

A) T = TRACK (!0 - !159), S = SECTOR (!0 - !10), O = OFFSET (!0 - !511)
B) S = SECTOR (!0 - !1760), O = OFFSET (!0 - !511)
EXAMPLE TO READ/DISPLAY ROOTBLOCK:
RT !75 !10
M T!80 OR M T!80S0 OR M T$50S0O0 OR M T50S0O0 OR M S!880 OR M S370

EDITOR-TOOLS:

HELP:	THIS SHORT HELP
SHIFT:	NO SCROLL/PAUSE
TAB:	INSERT SPACE(S)
ESC:	ESCAPES ANY COMMAND (NOT T/TS !)
F1:	CLR + HOME
F2:	HOME
F5:	PRINT SCREEN
F6:	SWITCH PRINTERDUMP ON/OFF
F7:	SWITCH OVERWRITE/INSERT MODE
F8:	SHOW INSTRUCTIONS OR MEMPEEKER
F9:	SWITCH GERMAN & US KEYBOARD
F10:	SWITCH SCREEN (BACK WITH SHIFT F10)

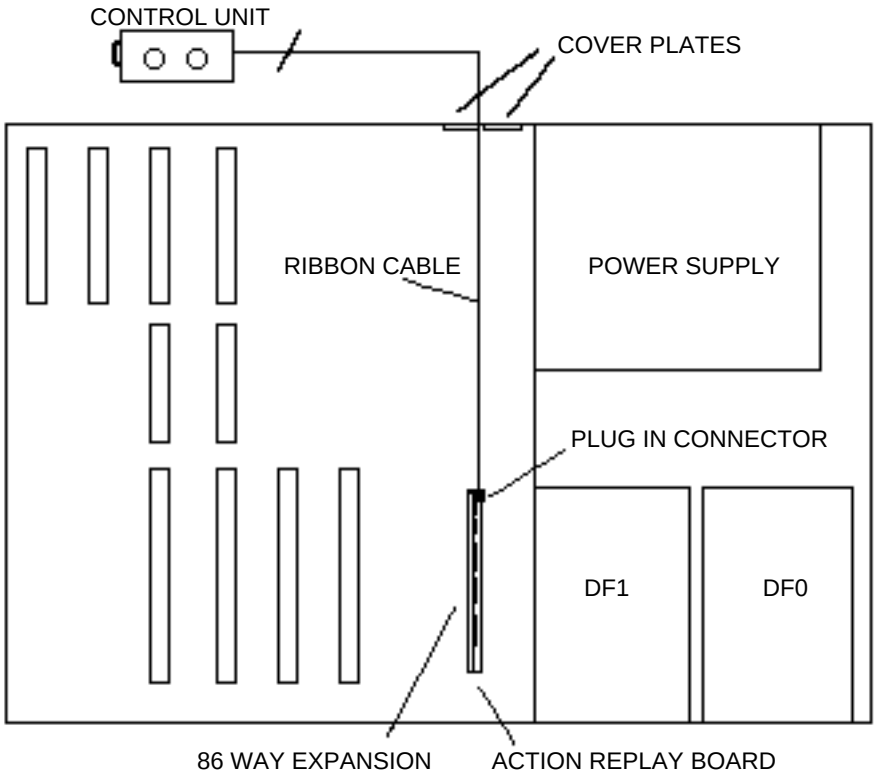
LED IS OFF = READY TO EXECUTE COMMANDS!
USE CURSORKEYS IN COMBINAION WITH SHIFT TOO

AMIGA 2000 INSTALLATION

The following text is for owners of the Amiga 2000 and is intended to replace the first paragraph of section 1 of the manual.

BEFORE PLUGGING OR UN-PLUGGING ANY DEVICE INTO YOUR AMIGA EXPANSION SLOTS YOU MUST ENSURE THE COMPUTER IS SWITCHED OFF. FAILURE TO OBSERVE THIS PRECAUTION WILL RESULT IN DAMAGE TO YOUR SYSTEM.

Firstly you must remove the case of your Amiga 2000. You will find a large number of expansion slots, the one you need is located towards the middle of the machine and is somewhat smaller than the others, it is an 86 way connector and the Action replay card should fix exactly in the slot with no spare connectors on either side. It should be inserted firmly with the extension lead connector facing towards the rear of the machine as in the diagram. The extension lead will then connect to the Action replay board, it will only fit in one way around, and should be lead out of the machine through an appropriate panel, one on the rear would probably suit best. Now replace the case of the machine. The control box can now be placed at an appropriate position by the side of the computer ready for freezing. To test the board simply power up the machine with the slommo switched off. When the startup screen (the hand with the disk) appears press the red freeze button. If all is well the screen should turn blue and the Action replay sign on menu will appear. If this does not happen switch your machine off and check the installation again. The manual should now be continued from page 4 starting from the section marked OPERATION.



FRONT OF AMIGA 2000

Addendum

SYSOP MODUS

Im erweiterten "SYS-OP"-Modus ist der Zugriff auf zuvor gesperrte Bereiche, wie nicht-autokonfigurierendes RAM und das ROM des Amiga Action Replay selbst möglich. Das Auslesen des ROMs ist interessant, da UAE (Amiga-Emulator für verschiedene Systeme) auch diese Hardware emulieren kann.

Um den SYS-OP-Modus freizuschalten geben Sie je nach Modell folgendes ein:
(RETURN nicht vergessen!)

Action Replay Mk 1	LORD OLAF
Action Replay Mk 2	MAY THE FORCE BE WITH YOU
Action Replay Mk 3	MAY THE FORCE BE WITH YOU Danach taucht folgende Meldung auf: "Try a new one" Geben Sie jetzt noch ein: NEW

Warnung: Unterbrechen Sie niemals während eines Schreib-/Lese-Zugriffs auf die Laufwerke (wenn die Laufwerks-LEDs leuchten). Daten auf dem Laufwerk können zerstört werden!

Kompabilität

Das Action Replay Mk2 funktioniert ausschließlich mit dem Prozessor M68000. Wenn eine Turbokarte im Prozessor-Sockel installiert ist, hängt sich der Amiga direkt beim Einschalten auf, wenn ein anderer Prozessor als der MC68000 startet.
Bei der Derringer-Turbokarte mit dem MC68030-Prozessor kann der MC68000 zusätzlich installiert werden. Installieren Sie einen Schalter an dem Jumper, der zwischen den Prozessoren umschaltet. Wenn Sie Ihren Amiga jetzt einschalten (Kaltstart), vergewissern Sie sich, daß der MC68000 ausgewählt ist. Nachdem der Amiga seinen Startvorgang beendet hat, wählen Sie den MC68030-Prozessor aus und führen einen Tastatur-RESET (Control + lAmiga + rAmiga = Warmstart) aus. Der Amiga startet jetzt auch mit dem MC68030. Natürlich funktioniert das Action Replay NICHT mit dem MC68030-Prozessor. Wenn Sie den Taster drücken, stürzt der Rechner ab!

Das Action Replay Mk2 funktioniert mit den KickStart-Roms 1.3 und 2.x. Mit dem KickStart-Rom 3.1 stürzt der Rechner beim Kaltstart ab.